Mise en oeuvre de l'agilité au sein de la société CosiWeb

Jean-Pierre Fontes

▶ **To cite this version:**

Jean-Pierre Fontes. Mise en oeuvre de l'agilité au sein de la société CosiWeb. Performance et fiabilité [cs.PF]. 2015. dumas-01586868

HAL Id: dumas-01586868

https://dumas.ccsd.cnrs.fr/dumas-01586868

Submitted on 13 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CONSERVATOIRE NATIONAL DES ARTS ET MÉTIERS

CENTRE RÉGIONAL ASSOCIÉ DE TOULOUSE

MÉMOIRE

Présenté en vue d'obtenir

le diplôme d'ingénieur C.N.A.M.

SPÉCIALITÉ : INFORMATIQUE

OPTION : SYSTÈMES D'INFORMATIONS

par

Jean-Pierre FONTES

Mise en œuvre de l'agilité au sein de la société CosiWeb

Soutenu le 22 mai 2015

JURY

PRESIDENT : **Yann POLLET** (Professeur, CNAM Paris)

MEMBRES : **Thierry MILLAN** (Responsable de la filière informatique, IPST CNAM)

Stéphane ISNARD (Ingénieur d'étude responsable SIRH, DSI UT2J)

Sébastien GUIBERT (Directeur administratif, CRCT INSERM)

Remerciements

Je tiens à remercier en premier lieu mon tuteur Thierry Millan, pour ses conseils, son soutien, et pour la relecture de ce mémoire. Je lui suis reconnaissant pour la disponibilité dont il a toujours fait preuve.

Je tiens à exprimer mes remerciements et ma gratitude à mon référent professionnel, Stéphane Isnard, pour son aide, ses conseils et sa disponibilité. Nos échanges m'ont permis à plusieurs reprises de trouver de nouvelles pistes dans le cadre de ce travail.

Je remercie le professeur Yann Pollet et Sébastien Guibert, pour avoir accepté de faire partie du jury de soutenance ainsi que tous les autres membres pour le temps consacré à la lecture de ce mémoire et pour me permettre de leur présenter le résultat de ce travail.

Je remercie tout spécialement ma compagne, Audrey, non seulement pour son aide et ses précieux conseils dans l'écriture de ce mémoire, mais aussi pour sa patience et les efforts qu'elle a accepté de consentir depuis 2009, date de ma première inscription au CNAM de Toulouse. Une douce pensée également pour ma fille Julia, née durant ce mémoire et qui n'a pas profité de son papa aussi longtemps qu'il l'aurait voulu.

J'aimerais, pour finir, remercier tous ceux qui m'ont aidé à un moment du projet, parmi lesquels mes parents, Sébastien Balard, Michaël Nepyjwoda, Jean-Michel Inglebert et bien entendu Vincent Tran-Van, mon associé et ami, qui a accepté de se lancer dans l'aventure agile avec tous les changements que cela a amenés pour notre entreprise.

Le parcours d'un auditeur du CNAM n'est pas le chemin d'une seule personne mais celui de la famille et des amis qui l'accompagnent.

Liste des abréviations

API : Application Programming Interface

ASIPAT : Application de Suivi Individuel de Projets Agiles et de Tests fonctionnels

BFR : Besoin en Fonds de Roulement

CLUSIF : Club de La Sécurité de l'Information Français

CMS : Content Management System

CRCT : Centre de Recherche en Cancérologie de Toulouse

DOM : Document Object Model

IDE : Integrated Development Environnement

IHM : Interaction Homme-Machine

LAMP : Linux, Apache, PHP, MySQL

PO : Product Owner

SGC : Système de Gestion de Contenu

SOAP : Simple Object Access Protocol

TDD : Test Driven Development

TIC : Technologies de l'Information et de la Communication

WAMP : Window, Apache, MySQL, PHP

WSDL : Web Services Description Language

XML : eXtensible Markup Language

XP : Extreme Programming

Glossaire

Burndown chart

Graphique de progression permettant de visualiser l'avancement des tâches réalisées au cours d'un sprint.

Daily Scrum meeting (mêlée quotidienne)

Réunion quotidienne de coordination de l'équipe Scrum d'une durée de quinze minutes.

Effet tunnel

Ce terme désigne un projet dans lequel le client ne communique pas avec l'entreprise durant la phase de réalisation et attend la fin du travail pour lui faire un retour. Le risque est une inadéquation entre les besoins du client et le produit ou le service livré.

Equipe Scrum

Equipe en charge du développement du produit. Elle est pluridisciplinaire et n'est pas hiérarchisée, les décisions sont prises de façon collégiale.

Portail web

Site web proposant des fonctionnalités adaptées à un domaine d'intérêt ou une communauté particulière. Le portail web guide l'internaute vers des services et des ressources liées à ses centres d'intérêt. Un portail web s'appuie sur une solution de CMS (*Content Management System* ou SGC pour *Système de Gestion de Contenu* en français).

Product-backlog (carnet de produit)

Liste toutes les exigences du product-owner à réaliser durant le projet et les classe par priorité.

Product-owner ou PO (propriétaire du produit)

Personne représentant le client ou les utilisateurs finaux.

Tests d'intégration

Tests réalisés par l'équipe sur l'ensemble du code pour valider que toutes les parties développées indépendamment fonctionnent bien ensemble.

Tests fonctionnels

Tests destinés à vérifier que toutes les exigences portant sur les fonctionnalités du système (fonctionnelles) soient satisfaites.

Tests unitaires

« *Un test unitaire est un test qui vérifie un comportement d'un morceau de code* » (Vernois, 2014). A chaque revue du code source, il permet aussi de vérifier que le résultat des modifications ne produit pas de régression (introduction de défauts) dans l'application.

Triangle de la performance

Figure constituée des trois sommets que sont la qualité (réponse aux besoins du client), le coût (respect du budget du projet) et les délais (respect du calendrier du projet). Dans une approche classique de gestion de projet en particulier, la satisfaction de ces contraintes mène à la satisfaction du client.

Scrum-master

Animateur d'équipe faisant la liaison entre le product-owner et l'équipe Scrum, il peut aussi faire partie de l'équipe et est en charge de la bonne mise en pratique de la méthode.

Sprint (itération)

Période de quelques semaines (un mois en général) à l'issue de laquelle une version partielle mais opérationnelle du produit est livrée.

Sprint-backlog (carnet de l'itération)

Liste des éléments du product-backlog à réaliser au cours d'une itération d'un projet agile.

Sprint planning meeting (réunion de planification d'itération)

Réunion ayant lieu avant chaque sprint et au cours de laquelle on sélectionne les user stories qui seront à développer.

Sprint review meeting (revue d'itération)

Réunion de fin de sprint dans laquelle le PO est convié afin de lui présenter la version partielle et fonctionnelle de son produit.

Task board

Tableau dans lequel sont listés à l'aide de post-it les fonctionnalités du sprint-backlog dans quatre colonnes représentant leur état d'avancement : à faire, en cours, à valider, fait.

User story (récit d'utilisateur)

Exprime une exigence du PO sous forme d'une phrase simple permettant de décrire une fonctionnalité à développer en répondant à trois interrogations : qui ? Quoi ? Pourquoi ? Voici un exemple : « *En tant qu'utilisateur, je veux pouvoir rechercher un étudiant par son numéro afin de le retrouver rapidement dans l'annuaire* ».

Table des matières

Introduction

Les méthodes agiles sont un sujet de débat qui revient fréquemment dans la communauté informatique et scientifique, mais aussi entre collègues au détour d'un couloir… Les convaincus, les sceptiques, ceux qui pratiquent, ceux qui pensent le faire et ceux qui s'y refusent ont souvent leur propre opinion de l'agilité. L'approche agile est citée par ses partisans comme étant la meilleure façon de gérer un projet et de collaborer au sein de l'équipe et avec les clients (Messager & Tabaka, 2010 ; Boisvert & Trudel, 2011 ; Fernandez, Houy, & Khalil, 2013). Hors, il est important pour l'entreprise CosiWeb, dont je suis le co-gérant, d'étudier toutes les solutions permettant d'améliorer notre compétitivité et la qualité de nos produits. Cela nous a conduits à nous demander si la mise en œuvre des pratiques agiles peut permettre de combler nos besoins actuels en matière de gestion de projet. Potentiellement, l'approche agile serait un vecteur d'amélioration du fonctionnement de l'entreprise en bénéficiant d'une plus grande adaptabilité avec nos clients, en apportant de la qualité logicielle, en ayant une meilleure visibilité sur nos projets et en réduisant le risque d'échec de ces derniers.

Déterminer si l'agilité peut effectivement améliorer la gestion de projet de CosiWeb implique un travail de recherche bibliographique et demande également l'analyse de sources plus directes comme des enquêtes auprès de nos clients et auprès d'experts des méthodes agiles, ainsi que d'assister à des conférences sur le sujet. L'exploitation de ces sources nous permettra de répondre à une série d'interrogations inhérentes au sujet : quels sont les besoins qui peuvent être comblés par les pratiques agiles ? Quelles sont les principales méthodes existantes et comment les mettre en œuvre ? Quelles sont les limites de l'agilité ?

Après une première partie consacrée à la présentation du contexte dans lequel évolue la société CosiWeb, nous mènerons une analyse de l'existant et des besoins méthodologiques actuels de cette dernière à l'aide d'une approche originale empruntée au domaine de la sécurisation informatique, la méthode Méhari. L'étude des principales méthodes de gestion de projet permettra ensuite de comparer les approches classiques et agiles et de voir en quoi cette dernière peut répondre en théorie aux besoins constatés.
La section suivante, relative à la mise en pratique de l'agilité, nous permettra de voir si la mise en œuvre des connaissances théoriques que nous avons acquises apporte des réponses à notre problématique initiale. Nous verrons ainsi que les pratiques agiles permettent de combler les

besoins que nous avons recensés, tant en matière de gestion de projet que d'ingénierie logicielle, en particulier dans l'élaboration des tests.

Enfin, la dernière partie offre une analyse des limites de l'agilité tout en fournissant, grâce aux enseignements que nous a appris cette étude, des préconisations à une entreprise qui souhaiterait adopter cette démarche. Ce mémoire tend ainsi à montrer que l'agilité peut être une intéressante alternative à une méthode de gestion de projet classique, dans la mesure où est menée une réflexion sur sa pertinence pour l'entreprise et sur les changements organisationnels qu'elle implique au sein de cette dernière.

Contexte

CosiWeb est née de la volonté d'entreprendre de Vincent Tran-Van et moi-même, Jean-Pierre Fontes. Cette aventure remonte au 30 juin 2006, date de la création de la société.
Nous allons présenter dans un premier temps l'entreprise CosiWeb et dans un second temps les marchés dans lesquels elle évolue.

1. Présentation de la société CosiWeb

Nous dresserons dans cette section la « *carte d'identité* » de l'entreprise. CosiWeb est spécialisé dans la conception de sites web pour le secteur public et privé et fait du « *business to business* » ou « *B2B* », c'est-à-dire qu'elle ne s'adresse qu'aux professionnels. Ce type d'entreprise est appelé communément une agence web ou « *web agency* ». L'entreprise est située sur Toulouse dans le quartier de Montaudran. CosiWeb est une société à responsabilité limitée (SARL) dont Vincent Tran-Van et moi-même sommes associés majoritaires (nous détenons le même nombre de parts) et gérants[1].

1.1. Domaine d'activité

Au lancement de CosiWeb, nous proposions dans nos services la maintenance informatique et la vente de sites web. Nous nous sommes rendu compte la première année que l'activité de maintenance n'était pas suffisamment rentable et nous nous sommes recentrés sur la vente de portails web.
Les portails web que nous proposons sont basés depuis 2006 sur un CMS propriétaire (*Content Management System* ou SGC pour *Système de Gestion de Contenu* en français) que j'ai entièrement conçu et qui se nomme CosiX. L'application ASIPAT ayant été développée dans le cadre de ce mémoire et que nous présenterons en page 75 est conçue à partir de CosiX. Nous pouvons aussi être amenés à commercialiser des solutions basées sur des CMS « *open-source* » si le client le souhaite.

1.2. Organisation interne de l'entreprise

CosiWeb est une petite entreprise constituée de deux gérants aux compétences complémentaires et dont la répartition des tâches est nette, comme le présente le schéma ci-dessous.

---

[1] Le dirigeant d'une SARL est dénommé « *gérant* ».

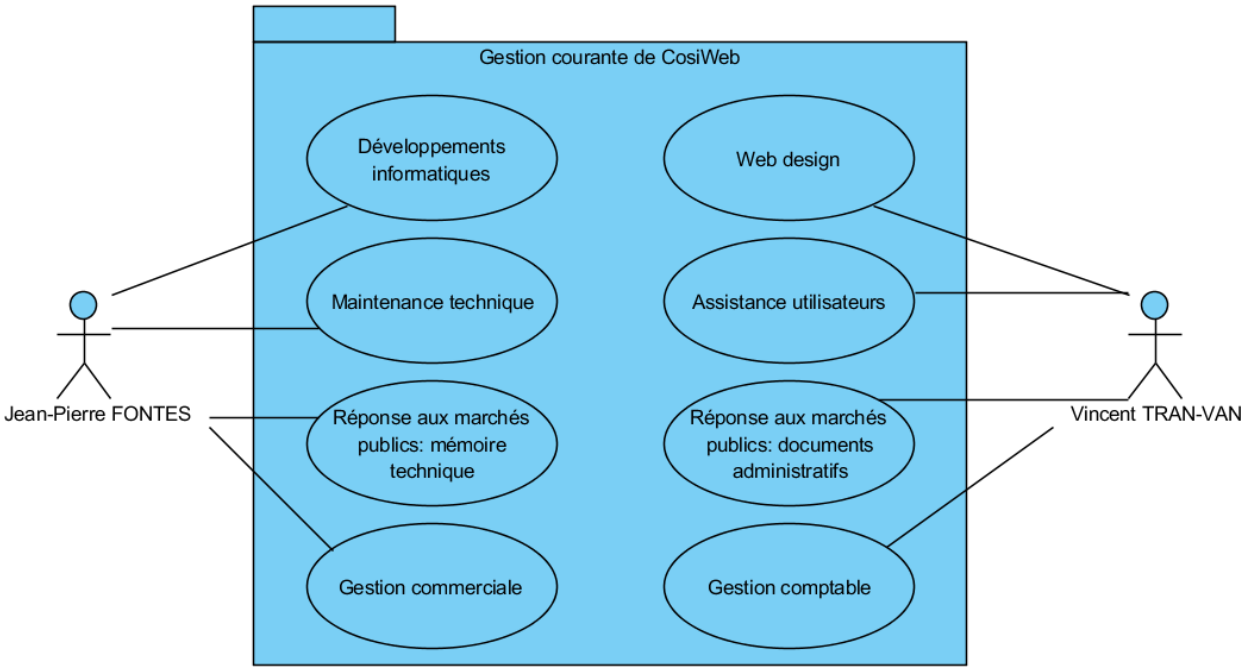


Figure 1 : Répartition des tâches au sein de l'entreprise CosiWeb entre les gérants.

L'entreprise peut faire appel à de la sous-traitance lorsque les travaux demandés sont hors de ses domaines de compétences, le cas ne se présente toutefois que pour de la rédaction et de la traduction de textes.

1.3. Evolution de l'activité

L'entreprise est en constante évolution depuis sa création avec un chiffre d'affaire et un résultat dont la tendance est à la hausse. Nous n'avons pas ressenti les effets de la crise de 2008, cela s'explique par la diversification de notre clientèle allant du commerçant au laboratoire de recherche et par le fait que nous nous adressons aux secteurs public et privé. Cela nous permet de réduire le risque de baisse de l'activité en nous recentrant sur des marchés plus porteurs ou stables comme le secteur public depuis 2008.

2. Marchés actuels de CosiWeb

Le marché est le lieu de rencontre entre l'offre et la demande d'un bien ou d'un service (Brotschi, 2011). CosiWeb évolue dans deux marchés distincts, celui des portails web depuis 2006 et celui beaucoup plus restreint des réseaux communautaires pour étudiants et anciens élèves depuis 2011.

2.1. Le marché des portails web

Concernant l'offre, **le marché est concurrentiel** ; en effet, sur la simple requête « *création de site web* » dans les pages jaunes et sur le secteur Toulouse et agglomération, nous pouvons recenser cent quarante-huit concurrents.

Concernant la demande, **le marché de CosiWeb est large**, car une agence web peut théoriquement proposer ses services pour tout type d'activités et pour des clients situés n'importe où dans le monde sans avoir à les rencontrer physiquement.

Nous évoluons donc dans un marché possédant de nombreux vendeurs et de nombreux acheteurs ; selon Stackelberg[2] (1934) nous sommes dans un **marché de concurrence pure et parfaite**. Cela se traduit par un marché stable puisqu'il n'y a pas d'acheteur suffisamment important pour avoir la capacité d'influencer les prix et la concurrence élevée empêche une envolée des tarifs.

L'activité de CosiWeb est concentrée sur les départements de la Haute-Garonne et du Tarn car nos clients recherchent un rapport de proximité et cela nous évite aussi des déplacements coûteux en temps. La demande n'en reste pas moins importante.

La situation géographique de CosiWeb est stratégique car la majorité de nos clients sont situés dans un rayon de vingt kilomètres autour de nos bureaux.

Nous proposons des sites web pour le secteur privé (les entreprises et les commerces) et le secteur public (essentiellement les mairies, l'enseignement supérieur et la recherche). En 2013[3], nous avons réalisé dix projets de sites web sur l'année.

---

[2] Le tableau de Stackelberg recense les différents types de concurrence que l'on peut rencontrer sur un marché.

[3] Lors de l'écriture de ce mémoire, le bilan comptable 2014 n'ayant pas été arrêté, nous nous baserons sur les chiffres de 2013.

Voici l'évolution du nombre de nouveaux clients cumulés de l'entreprise depuis 2006 :

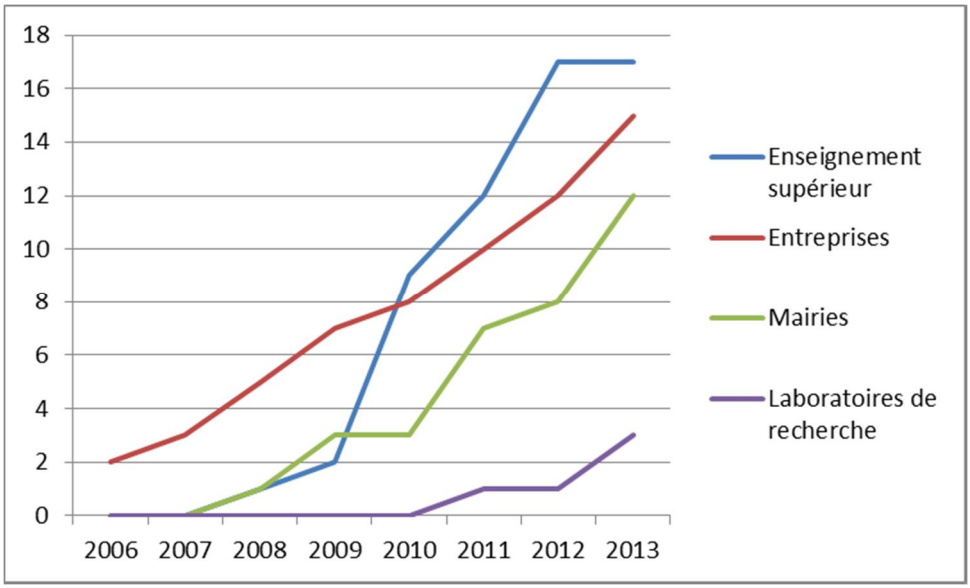


Figure 2 : Évolution depuis 2006 du nombre de nouveaux clients cumulés de CosiWeb classés par type.

La nette augmentation des clients de l'enseignement supérieur s'explique par un effet « *boule de neige* » lorsque nous avons réalisé le site de l'IUT A de Toulouse qui nous a amené d'autres contrats par la suite. La répartition de la clientèle de CosiWeb en 2013 en fonction du chiffre d'affaire généré se présente de la sorte :

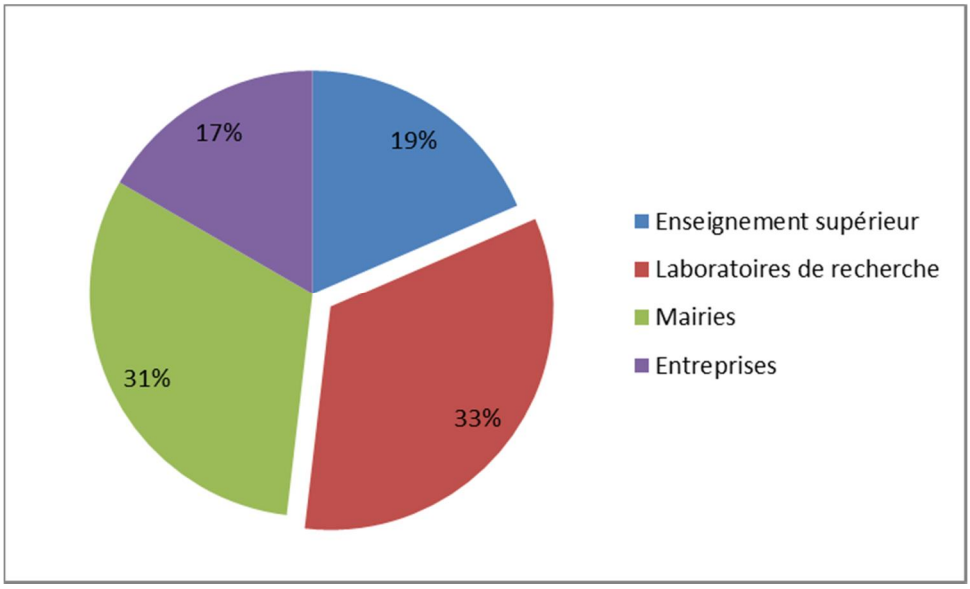


Figure 3 : Répartition de la clientèle de CosiWeb en fonction du chiffre d'affaire généré en 2013.

On peut voir que 83% de la clientèle de CosiWeb se situe dans le public et 17% dans le privé. C'est un choix stratégique dû au fait que le secteur public présente moins d'aléas dans le paiement de notre travail, ce qui nous permet de limiter notre besoin en fonds de roulement[4].

2.2. Le marché des réseaux communautaires pour étudiants actuels et anciens

Dans le marché des réseaux communautaires, sur lequel nous sommes positionnés avec notre solution LiveRezo depuis 2011, la concurrence est constituée de l'ensemble des entreprises qui créent des réseaux communautaires pour étudiants et la demande, par les établissements de l'enseignement supérieur.

Nous pouvons écarter de notre concurrence les réseaux communautaires populaires comme Facebook, Viadeo et LinkedIn, car ces derniers ne sont pas aussi spécialisés que LiveRezo dans les fonctionnalités offertes pour les étudiants et n'offrent pas d'outils dédiés aux formations. Par ailleurs, leur modèle économique est différent de celui conçu pour LiveRezo, ce dernier est en effet basé sur un abonnement auquel souscrivent les formations de l'enseignement supérieur.

Nous nous situons ainsi dans un marché de niche, **l'offre étant très faible**.

La demande est quant à elle concentrée sur les formations de l'enseignement supérieur dans l'ensemble du territoire français et représente un **marché important**.

Ce marché possède donc une offre très faible et de nombreux acheteurs ; selon Stackelberg (1934) nous sommes dans un **marché oligopolistique**. Cette situation est avantageuse car la demande est plus importante que l'offre, cela permet de mener une politique tarifaire moins soumise à la pression de la concurrence et d'augmenter la marge et par conséquent la rentabilité du produit.

Sur ce marché, l'activité de CosiWeb est concentrée sur la région Midi-Pyrénées et plus particulièrement sur la région toulousaine. Toulouse étant une métropole avec trois grandes universités, la demande est importante.

Actuellement, notre clientèle est concentrée sur les départements de l'IUT A Toulouse, Auch et Castres et nous commençons à travailler avec l'université Toulouse Jean Jaurès par l'intermédiaire de l'ISTHIA (Institut supérieur du tourisme, de l'hôtellerie et de l'alimentation).

Cela représente quatorze établissements et quarante-trois diplômes gérés en 2015.

---

[4] Selon Brotschi (2011), le besoin en fonds de roulement (BFR) est le décalage de trésorerie provenant de l'activité courante de l'entreprise. Les créances des clients font augmenter le BFR. Une entreprise ayant remporté suffisamment de contrats pour son activité peut faire faillite si son BFR devient trop important.

3. Résumé et perspectives

Cosiweb est une SARL créée en 2006, dirigée par deux gérants aux compétences complémentaires. L'entreprise a conçu sa propre solution de CMS appelée CosiX et qui sert de base technique à ses deux principaux produits : les portails web et LiveRezo, son réseau communautaire pour étudiants et anciens étudiants.

Le marché des portails web est très concurrentiel et laisse peu de latitude à la politique tarifaire de l'entreprise, par ailleurs l'évolution technologique rapide des CMS Open-source incite CosiWeb à améliorer sans cesse son produit pour éviter le risque d'obsolescence. Il devient nécessaire que CosiWeb adopte une démarche structurée et rigoureuse dans sa gestion de projet afin d'améliorer la qualité logicielle de son CMS, réduire les délais de mise en production et tenter de baisser les coûts de développement. Pour ce faire, nous allons à présent analyser la gestion de projet actuelle dans l'entreprise afin de déceler les dysfonctionnements existants et d'en extraire les besoins méthodologiques qui en découlent.

Analyse de l'existant et des besoins méthodologiques actuels

L'objectif de cette analyse de l'existant est de mettre en avant les dysfonctionnements éventuels dans la méthode actuelle de gestion de projet de l'entreprise et les besoins qui en découlent.
Cette analyse s'est déroulée en trois étapes :

- Nous avons tout d'abord tenté de relever et de classer les dysfonctionnements qu'il pouvait y avoir dans la gestion de projet actuelle.
- Les dysfonctionnements repérés peuvent avoir un impact insignifiant ou au contraire important dans la vie d'un projet, nous avons cherché une méthode nous permettant d'évaluer et de prioriser ces dysfonctionnements.
- Suite aux résultats de la précédente étape, nous avons tenté de déterminer les besoins de l'entreprise.

Avant de présenter cela, nous définirons ce qu'est un projet et ferons une courte présentation de la gestion de projet actuelle à CosiWeb.

1. Préambule

1.1. Qu'est-ce qu'un projet ?

Nous allons aborder les termes de « *projet* » et de « *gestion de projet* » durant tout ce mémoire, il est donc important de définir ces notions.
Selon Messager (2010) qui reprend une définition du Project Management Institute[5], « *un projet est une entreprise* ***temporaire*** *décidée dans le but de créer* ***un produit, un service ou un résultat unique*** ».
Un projet est **temporaire**, car il possède un début et une fin, la fin marquant l'atteinte des objectifs ou au contraire la constatation de l'échec du projet.
L'objectif de tout projet est la création d'un **produit** ou d'un **service** et le résultat de chaque projet est **unique**, même si comme le souligne Messager (2010), des éléments peuvent être reproductibles ou réutilisables.

[5] http://www.pmi.org, le PMI est une organisation internationale de standardisation du management de projet.

La gestion de projet est la mise en œuvre d'une méthode permettant d'atteindre l'objectif final qui est la création du produit ou du service, tout en prenant en compte des contraintes de qualité, de coût et de délai.

Ces trois variables constituent le **triangle de la performance**, selon Zorzabalbère (2011) :

- La **qualité** est la réponse au besoin du client.
- Le **coût** est le budget nécessaire à la mise en œuvre du projet.
- Le **délai** de réalisation est le calendrier dans lequel le projet doit se réaliser.

Figure 4 : Triangle de la performance.

La satisfaction de ces contraintes mène à la satisfaction du client. Ainsi, le chef de projet doit avoir comme souci permanent le respect de triangle de la performance. Cela est générateur de stress et de tensions éventuelles avec le client car lorsque ce dernier valide une étape du projet, CosiWeb ne peut théoriquement plus revenir dessus. Cela se traduit par des fonctionnalités qui sont développées mais qui peuvent ne plus correspondre aux besoins du client et qui font quitter le projet du périmètre du triangle de la performance par une baisse de qualité. Si au contraire l'entreprise décide de revoir une fonctionnalité validée, elle prend le risque de sortir du triangle de la performance en dépassant les délais ou le coût de réalisation. Un des enjeux du travail effectué dans le cadre de ce mémoire consiste donc à faciliter et à rendre plus sereine la gestion de projet actuelle, tout en atteignant la satisfaction du client.

1.2. Méthode actuelle de gestion des projets à CosiWeb

La gestion de projet actuelle de CosiWeb s'appuie sur un cycle en cascade (voir page 39) dont voici les principales caractéristiques :

- Le cycle de vie du projet est constitué de phases séquentielles : recueil des besoins, spécifications, codage...

- On ne peut passer à la phase suivante que lorsque les livrables de la phase actuelle sont validés.
- On ne peut pas revenir à la phase précédente une fois celle-ci validée.
- Tous les besoins du projet sont exprimés et recueillis lors de la première phase.
- Le système est conceptualisé et validé avant la phase de développement.

Nous allons maintenant présenter les problèmes que rencontre CosiWeb dans cette gestion de projet.

2. Recensement et classification des dysfonctionnements existants

Nous considérons comme un dysfonctionnement toute activité ayant déjà eu un impact négatif sur l'entreprise et/ou le client. Un impact négatif peut être un dépassement de délai, de coût et/ou une dégradation de la qualité du produit.

2.1. Recensement des dysfonctionnements existants

Pour des raisons d'objectivité et de prise de recul, il nous a paru indispensable qu'une personne extérieure à CosiWeb relève les dysfonctionnements éventuels au sein de l'entreprise. Nous avons donc fait appel à un étudiant (Jean-Baptiste Vazquez) de licence 3 en management des entreprises en réseau de l'Université Paul Sabatier, pour un stage de huit semaines.
Les missions que j'avais définies pour ce stage étaient les suivantes :

- Audit des besoins méthodologiques de l'entreprise (durée de quatre semaines).
- Audit des besoins juridiques de l'entreprise (durée de deux semaines).
- Médiatisation de sites web et communication (durée de deux semaines).

Seule la première mission concerne ce mémoire.
J'ai laissé l'étudiant me proposer une approche permettant de mener à bien cet audit des besoins méthodologiques tout en fixant le périmètre suivant :

- Se focaliser sur les dysfonctionnements qui peuvent exister d'un point de vue organisationnel (dans la gestion du projet en interne et avec le client) et technique (dans la manière de concevoir nos produits).
- Mettre de côté les aspects liés à la gestion comptable et financière de l'entreprise. En effet, cette dimension n'est pas du ressort d'un ingénieur en informatique et sort du périmètre dans lequel se déroule notre étude.

Après une semaine de travail en autonomie et des échanges quant à la méthode la plus adaptée pour recueillir les dysfonctionnements, l'étudiant a réalisé un entretien avec nous (les co-gérants) en janvier 2014. Les questions portaient sur la gestion de projet actuelle de CosiWeb et sur les points suivants en particulier :

- Le recueil des besoins.
- La planification.
- La communication dans l'équipe et avec le client.
- Le suivi des projets.
- Les tests.
- La validation des travaux.

Je me suis ensuite chargé de l'analyse des réponses données dans une volonté de structuration et d'approfondissement.
Nous allons présenter la façon dont nous avons classé les dysfonctionnements mis en avant lors de cet entretien.

2.2. Classification des dysfonctionnements constatés

Selon Zorzabalbère (2011), un projet est composé de trois principales phases : préparation, pilotage et progression. Ces phases s'apparentent à un cycle classique de gestion de projet, dit en « *cascade* » (voir page 39). Etant donné que ce cycle de vie d'un projet était suivi jusqu'à présent par CosiWeb, nous allons classer dans ces différentes phases l'ensemble des dysfonctionnements constatés lors des entretiens, tout en les détaillant. Cela nous servira par la suite à déterminer leurs conséquences potentielles sur l'entreprise.

2.2.1. Phase de préparation

2.2.1.1. Formulation des besoins du client

Dans les projets menés avec des clients du secteur privé, ces derniers fournissent rarement un cahier des charges et quand c'est le cas, il est souvent très succinct. L'entreprise mène un seul entretien qui en général ne suffit pas à comprendre l'ensemble des besoins du client, besoins évoluant souvent par la suite. En effet, cet entretien commercial se fait avant la contractualisation et représente du temps et par conséquent un coût pour l'entreprise.
Le problème se pose moins dans le cadre de marchés publics, car le cahier des charges technique fourni est généralement détaillé ; il sera contractuellement respecté par les deux parties.

Dysfonctionnements : pas de prise en compte de tous les besoins du client, risque de fournir un produit inadapté à la demande.

2.2.1.2. Recueil des besoins du client

Au lancement du projet, une première réunion est conduite par l'entreprise afin de cerner plus précisément la demande du client. L'entreprise ne reformule pas systématiquement les besoins du client et ne produit pas toujours de compte rendu de réunion.
Dysfonctionnements : mauvaise compréhension des besoins du client, risque de fournir un produit inadapté à la demande.

2.2.1.3. Planification

L'entreprise n'a pas de stratégie de recensement et de priorisation des tâches à réaliser. Elle ne produit pas toujours de rétro-planning qui permettrait d'avoir une vision chronologique du projet. Quand un planning est réalisé, il n'est plus mis à jour une fois que le projet est lancé.
Dysfonctionnements : pas de priorité dans les développements, dépassement des délais.

2.2.1.4. Validation du cadrage du projet

La validation du cadrage du projet consiste à faire valider par le client les fonctionnalités attendues et le rétro-planning quand celui-ci est produit. Cette validation n'est pas toujours réalisée par l'entreprise ou de façon informelle (orale).
Dysfonctionnements : le client peut demander des modifications incessantes sur le produit.

2.2.2. Phase de pilotage

2.2.2.1. Conception du produit

Les tests unitaires sont inexistants. Il n'y a pas de tests d'intégration car il n'y a qu'un seul développeur au sein de CosiWeb.
Les tests fonctionnels sont réalisés manuellement, en observant à l'écran le résultat de ses modifications, mais sans méthode particulière et sans outil dédié.
Dysfonctionnements : constatation tardive de bogues. Régression possible lors des modifications du code source.

2.2.2.2. Communication au sein de l'entreprise

Il y a un manque de rigueur dans l'élaboration des comptes rendus, dans la prise en compte des retours des clients et un manque de communication entre les deux gérants de l'entreprise.
Dysfonctionnements : problème de communication entre les gérants, les retours clients ne sont pas toujours formalisés et relayés avec exactitude. Cela peut provoquer une perte d'information dans les projets.

2.2.2.3. Communication avec le client

Il y a une réelle difficulté à impliquer le client dans la phase de pilotage. Il est souvent difficile d'obtenir des retours de sa part, cela se traduit par des écarts dans les résultats attendus durant la phase suivante de progression.
Dysfonctionnements : on rencontre un effet tunnel sur certains projets. Le produit ne répond pas correctement aux besoins du client.

2.2.2.4. Suivi du projet

Le client n'est pas toujours au courant de l'état d'avancement du projet.
Il n'existe aucun reporting permettant de suivre et de mesurer les écarts éventuels entre les fonctionnalités réalisées par l'entreprise et celles attendues par le client.
Dysfonctionnements : dépassements de délais et mauvaise priorisation des tâches à réaliser.

2.2.3. Phase de progression

2.2.3.1. Validation des travaux

Il n'y a pas toujours de validation des fonctionnalités développées.
Dysfonctionnements : demandes incessantes du client après déploiement du produit.

2.2.3.2. Bilan du projet

L'entreprise ne recense pas les retours client écrits ou oraux à l'issue de la phase de préparation et de pilotage. L'entreprise ne fait pas de rétrospective critique de la gestion du projet à l'issue des travaux.
Dysfonctionnements : risque de répéter les mêmes erreurs.

2.3. Comment analyser ces dysfonctionnements ?

Maintenant que nous avons détaillé les dysfonctionnements constatés, il s'agit d'estimer les risques qu'ils font courir à l'entreprise et aux clients ainsi que leur gravité. Cela nous permettra de cibler les dysfonctionnements les plus problématiques.
Nous avons choisi pour cela d'employer une méthode d'analyse des risques étudiée au CNAM (Ayari, 2013) et utilisée habituellement dans la sécurité informatique, la méthode Méhari. Nous allons voir en quoi cette méthode est pertinente pour répondre à notre problématique.

3. Evaluation des dysfonctionnements avec la méthode Méhari

3.1. Pourquoi choisir la méthode Méhari ?

La problématique que nous nous sommes posés est de savoir comment mesurer et prioriser les dysfonctionnements constatés selon leur gravité pour l'entreprise.
Méhari propose une méthode et des outils permettant d'objectiver le plus possible les dysfonctionnements en les quantifiant de manière précise.
C'est une méthode reconnue et utilisée dans de grandes entreprises comme AIRBUS[6]. Elle est conforme aux exigences de la norme ISO/IEC 27005 décrivant le système de management des risques liés à la sécurité de l'information.
Par ailleurs, nous avons eu l'occasion de nous familiariser avec cette méthode au sein du CNAM dans l'enseignement RSX112 (Ayari, 2013).

3.2. Qu'est-ce que la méthode Méhari ?

Méhari est une méthode d'analyse et de gestion des risques liés à l'information. Elle est conçue et maintenue par l'association CLUSIF[7] (club de la sécurité de l'information français) et son utilisation est gratuite. Bien que cette méthode soit habituellement utilisée dans la sécurisation informatique (protection des actes malveillants), nous pensons qu'elle peut être transposée à une entreprise.
En effet, comme un système informatique, l'entreprise est un système d'information. Selon Batatia (2013) l'entreprise est un système qui est « *un ensemble de composants, structurés avec*

---

[6] Méhari est utilisé par les enseignants professionnels du cours du CNAM RSX112, dans le cadre de leurs projets pour AIRBUS.

[7] http://www.clusif.asso.fr/fr/production/mehari/#risques

des liens. Ces composants interagissent pour réaliser un but commun dans un environnement donné ».

Le système est soumis à des **menaces** (causes potentielles d'incidents) qui peuvent provoquer des dommages. Dans le cadre de notre étude, ces menaces sont les dysfonctionnements constatés.

La méthode Méhari va nous permettre d'évaluer la **gravité des risques** que font peser les dysfonctionnements constatés sur l'entreprise. Le risque est fonction (et non le produit) de la **potentialité** et de l'**impact.** La potentialité est la possibilité qu'un dysfonctionnement survienne et l'impact est la conséquence de ce dysfonctionnement sur l'entreprise. Nous allons maintenant expliquer comment mettre en œuvre Méhari dans le cadre de notre étude.

3.3. Les étapes de la méthode Méhari

Méhari permet dans un premier temps d'évaluer le niveau de risque en l'absence de mesures permettant de le réduire, on qualifie ce risque d' « ***intrinsèque*** » ; il en est de même pour la potentialité et l'impact, dits intrinsèques.

Dans un second temps, Méhari permet d'évaluer le risque « ***résiduel*** », c'est-à-dire le risque qui subsiste suite à l'adoption de mesures de sécurité ayant un effet dissuasif (identifier l'auteur de la malveillance), préventif (interdire une action), palliatif (limiter la propagation d'une erreur) ou de confinement (assurer la continuité des services). Dans le cas de notre étude, nous appliquons Méhari à une entreprise et les actes malveillants sont hors contexte en ce qui concerne la gestion de projet. Ainsi, nous ne suivrons pas les étapes de la méthode Méhari qui permettent d'évaluer un niveau de risque résiduel et qui sont hors sujet dans notre contexte.

Nous nous servirons donc de cette méthode comme d'un outil de diagnostic pour mettre en avant les dysfonctionnements présentant un niveau de gravité inacceptable pour l'entreprise.

Voici schématiquement la mise en œuvre de la méthode Méhari dans notre cas d'étude :

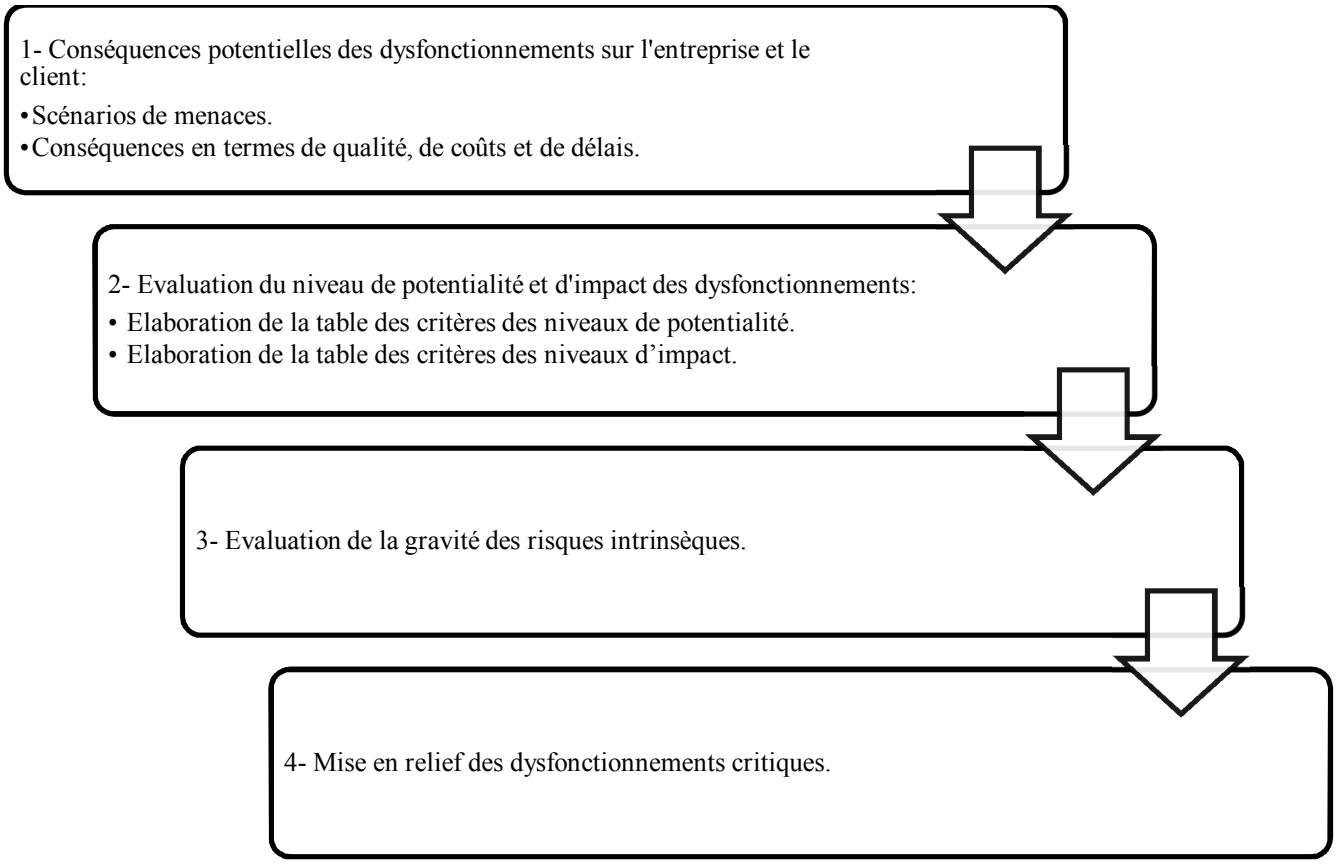


Figure 5 : Déroulement de la méthode Méhari appliquée à l'évaluation des dysfonctionnements dans la gestion de projet actuelle de CosiWeb.

Nous allons à présent détailler les étapes de la méthode Méhari.

3.3.1. Scénarios de menaces et conséquences potentielles des dysfonctionnements

Les problèmes constatés ont un impact sur l'entreprise et le client. Pour chaque dysfonctionnement, la méthode Méhari propose de décrire un **scénario** qui peut survenir entre l'entreprise et le client. Prenons l'exemple suivant :

Dysfonctionnement constaté dans le recueil des besoins du client :
« *Pas de prise en compte de tous les besoins du client.* »

Conséquence possible pour l'entreprise :
« *Des fonctionnalités à redévelopper.* »

Conséquence possible pour le client :

« *Le produit ne répond pas totalement au besoin.* »

Il faut ensuite déterminer l'incidence de ce scénario sur la performance de l'entreprise en termes de qualité, de coût et de délais. Ces trois variables interagissent entre elles et constituent le « ***triangle de la performance*** » que nous avons déjà abordé en page 19.

Un dysfonctionnement peut avoir un impact sur tout ou partie des trois variables.

Dans cet exemple, le dysfonctionnement a une incidence sur la qualité, le coût et les délais :

- La qualité, car le produit ne répondra pas exactement au besoin du client.
- Le coût, à cause des fonctionnalités à redévelopper.
- Le délai, à cause du temps supplémentaire et non prévu lié aux fonctionnalités à redévelopper.

Cela se traduira sous forme de tableau de la manière suivante :

Etape du projet	Dysfonctionnement constaté	Conséquences sur l'entreprise	Conséquences sur le client	Qualité	Coûts	Délais
Formulation des besoins du client	Pas de prise en compte de tous les besoins du client.	Développements imprévus.	Le produit ne répond pas totalement au besoin.	X	X	X

Nous allons faire de même pour chacun des dysfonctionnements constatés.

Ce travail va nous permettre de déterminer un niveau de risque que font peser ces problèmes sur l'entreprise en évaluant tout d'abord un niveau de potentialité et d'impact.

3.3.2. Evaluation du niveau de potentialité et d'impact des dysfonctionnements

Comme nous l'avons vu précédemment, le couple potentialité et impact constitue le risque.

Nous devons donc tout d'abord évaluer la potentialité d'occurrence des dysfonctionnements et leur impact sur l'entreprise. Nous tentons ainsi de répondre aux questions :

« *Actuellement à CosiWeb, quelle est la potentialité d'un dysfonctionnement ?* »

« *Actuellement à CosiWeb, quel est le niveau d'impact de ce dysfonctionnement ?* »

Nous allons concevoir deux tableaux dans lesquels nous définissons nos propres critères. Les critères des niveaux de potentialité et d'impact doivent être exhaustifs et englober l'ensemble du

périmètre des possibles. Nous nous sommes basé sur notre expérience et sur des cas pratiques d'emploi de la méthode Méhari afin de les élaborer.

3.3.2.1. Elaboration de la table des critères des niveaux de potentialité

Pour déterminer les critères de potentialité, il faut donner un nombre d'occurrences qu'un dysfonctionnement survienne sur une période donnée.
Pour la période, nous avons choisi l'année calendaire passée (2013) car elle est représentative de l'activité actuelle de CosiWeb en termes de types de clientèle et de nombre de projets réalisés (10 projets de création de sites web sur l'année).

	Niveau de potentialité	**Critères**
4	Très probable	S'est produit sur plus de 5 projets en 2013
3	Probable	S'est produit sur 2 à 5 projets en 2013
2	Peu probable	S'est produit sur 1 projet en 2013
1	Improbable	Ne s'est pas produit en 2013

Tableau I : Table des critères d'estimation de la potentialité d'occurrence des dysfonctionnements.

3.3.2.2. Elaboration de la table des critères des niveaux d'impact

Pour déterminer les critères des niveaux d'impacts, nous regardons si seule l'entreprise est touchée par un dysfonctionnement ou si le client l'est aussi. Puis nous décrivons l'impact sur la confiance du client et les conséquences sur le triangle de la performance de l'entreprise : qualité, coûts et délais.

	Niveau d'impact	**Critères**
4	Très élevé	• Impacte l'entreprise et le client • Confiance du client perdue • Produit inutilisable, conséquences importantes en termes de qualité, coûts et délais
3	Elevé	• Impacte l'entreprise et le client • Confiance du client fortement dégradée • Conséquences importantes en termes de qualité et/ou coûts et/ou délais
2	Moyen	• Impacte l'entreprise et le client • Confiance du client légèrement dégradée

		• Conséquences significatives en termes de qualité et/ou coûts et/ou délais
1	Faible	• Impacte uniquement l'entreprise • Sans conséquences sur la confiance du client • Conséquences faibles pour l'entreprise en termes de qualité et/ou coûts et/ou délai

Tableau II : Table des critères des niveaux d'impacts des dysfonctionnements sur l'entreprise.

3.3.3. Evaluation de la gravité des risques intrinsèques

L'évaluation du risque est obtenue grâce à la table de gravité globale des risques fournie par la version 2.14 de Méhari[8]. Les valeurs sont fixes et ne peuvent être modifiées. Pour obtenir le niveau de risque, il suffit de croiser le niveau d'impact et de potentialité (déterminés grâce aux tableaux élaborés précédemment).

Cette grille permet aussi de mettre en avant la zone des risques inacceptables indiquée dans les cellules rouges et orange. Un risque de niveaux 3 ou 4 est jugé inacceptable.

IMPACT

4	2	3	4	4
3	2	3	3	4
2	1	2	2	3
1	1	1	1	2
	1	**2**	**3**	**4**

POTENTIALITE

Tableau III : Table de gravité globale des risques de la méthode Méhari.

Si nous reprenons l'exemple précédent, nous pouvons déterminer le risque intrinsèque d'un dysfonctionnement et donc sa gravité :

Etape du projet	Potentialité intrinsèque	Impact intrinsèque	Risque intrinsèque (gravité)
Formulation des besoins du client	2	4	**3**

[8] http://www.clusif.asso.fr/fr/production/mehari/

Nous allons maintenant mettre en application la méthode Méhari afin de faire émerger les dysfonctionnements en termes de gestion de projet qui font courir un risque inacceptable à l'entreprise.

3.4. Mise en application de la méthode Méhari

Voici les différentes étapes de la méthode Méhari vues précédemment :

1. Elaboration de scénarios de menaces et conséquences potentielles des dysfonctionnements.
2. Evaluation du niveau de potentialité et d'impact de ces dysfonctionnements.
3. Evaluation de la gravité des risques intrinsèques liés aux dysfonctionnements.

Nous allons mettre en œuvre ces étapes afin de faire émerger les dysfonctionnements présentant **un niveau de gravité inacceptable (c'est-à-dire un risque de 3 ou 4)**. Le risque intrinsèque est présenté en dernière colonne.

Etape du projet phase de préparation phase de pilotage phase de progression	Dysfonctionnement constaté	Conséquences sur l'entreprise	Conséquences sur le client	Qualité *Respect des besoins du client*	Coûts	Délais	Potentialité intrinsèque	Impact intrinsèque	Risque intrinsèque (gravité)
Formulation des besoins du client	Pas de prise en compte de tous les besoins du client.	- Développements imprévus.	- Le produit ne répond pas totalement au besoin.	X	X	X	2	4	**3**
Recueil des besoins du client	Mauvaise compréhension des besoins du client.	- Mauvaise vision du projet. - Fonctionnalités à redévelopper.	- Le produit ne répond pas totalement au besoin.	X	X	X	2	4	**3**
Planification	Pas de priorisation des développements.	- Tensions avec le client.	- Les fonctionnalités importantes ne sont pas forcément livrées en priorité.			X	4	2	**3**
Planification	Dépassements des délais.	- Perte de temps sur le projet. - Pénalités éventuelles de retard.	- Date de livraison repoussée.		X	X	2	3	**3**
Validation du cadrage du projet	demandes incessantes de modifications du produit.	- Développements imprévus.		X			1	3	**2**
Conception du produit	Pas de *tests unitaires*, constatation tardive de bogues, régression.	- Correction chronophage et tardive des bogues. - Régressions possible de l'application.	- Risques de bogues et de dysfonctionnements du produit.	X	X	X	2	4	**3**
Conception du produit	*Tests fonctionnels* réalisés manuellement	- Correction chronophage et	- Risques de bogues et de dysfonctionnements	X	X	X	2	4	**3**

	et incomplets. Constatation tardive de bogues.	tardive des bogues.	du produit.					
Communication au sein de l'entreprise	Les retours clients ne sont pas toujours formalisés et relayés entre les gérants de l'entreprise. Perte d'information.	- Pas de vision globale du projet. - Risque de dépassement des délais. - Manque de cohésion et d'implication.	- Réunions intempestives. - Manque de réactivité de l'entreprise aux sollicitations. - Oublis dans la prise en compte des demandes.	X		X	3	3	**3**
Communication avec le client	Effet tunnel sur certains projets.	- Perte de temps à développer des fonctionnalités inadéquates.	- Vision floue du projet. - Inadéquation du produit aux besoins. - Peu d'implication dans le projet.	X		X	3	3	**3**
Suivi du projet	Le planning prévisionnel n'est plus mis à jour une fois le projet en phase de pilotage.	- Inadéquation avec les besoins du client.	- Vision floue du projet.	X		X	3	3	**3**
Suivi du projet	Absence de reporting de pilotage permettant de mesurer l'écart entre le réalisé et l'attendu.	- Pas de vision sur l'état d'avancement du projet. - Mauvaise priorisation des développements.	- Pas de vision sur l'état d'avancement du projet. - Date de livraison repoussée.			X	4	3	**4**

Validation des travaux	Pas de validation des fonctionnalités développées. Demandes incessantes du client après le déploiement du produit.	- Redéveloppements.	- Ecart par rapport aux demandes.		X	X	X	2	3	**3**
Bilan du projet	Pas de rétrospective critique en fin de projet. Risque de répéter les mêmes erreurs.	- Pas de retours d'expérience. - Mauvaise compréhension des problèmes rencontrés.			X			3	1	**1**

Tableau IV : Synthèse des résultats de la mise en œuvre de la méthode Méhari.

3.5. Mise en relief des dysfonctionnements critiques

Voici les dysfonctionnements présentant un niveau de gravité inacceptable dans la conduite d'un projet actuel au sein de CosiWeb (c'est-à-dire présentant un risque intrinsèque de 3 ou 4).

Etape du projet phase de préparation phase de pilotage phase de progression	Dysfonctionnement constaté	Risque intrinsèque (gravité)
Formulation des besoins du client	Pas de prise en compte de tous les besoins du client.	**3**
Recueil des besoins du client	Mauvaise compréhension des besoins du client.	**3**
Planification	Pas de priorisation des développements.	**3**
Planification	Dépassements des délais.	**3**
Conception du produit	Pas de *tests unitaires*, constatation tardive de bogues, régression.	**3**
Conception du produit	*Tests fonctionnels* réalisés manuellement et incomplets. Constatation tardive de bogues.	**3**
Communication au sein de l'entreprise	Les retours clients ne sont pas toujours formalisés et relayés entre les gérants de l'entreprise. Perte d'information.	**3**
Communication avec le client	Effet tunnel sur certains projets.	**3**
Suivi du projet	Le planning prévisionnel n'est plus mis à jour une fois le projet en phase de pilotage.	**3**
Suivi du projet	Absence de reporting de pilotage permettant de mesurer l'écart entre le réalisé et l'attendu.	**4**
Validation des travaux	Pas de validation des fonctionnalités développées. Demandes incessantes du client après le déploiement du produit.	**3**

Tableau V : Dysfonctionnements présentant un niveau de gravité inacceptable en termes de gestion de projet.

Au vu des dysfonctionnements constatés, quels sont les besoins actuels de CosiWeb qui permettraient de les réduire ?

4. Les besoins actuels de CosiWeb

Suite à l'analyse de l'existant, nous allons pouvoir faire émerger les besoins de l'entreprise. Nous listons pour cela des dysfonctionnements constatés et présentant un niveau de gravité inacceptable.

Dysfonctionnement constaté	Type de besoin
Pas de prise en compte de tous les besoins du client.	**S'adapter aux besoins du client**
Mauvaise compréhension des besoins du client.	**S'adapter aux besoins du client**
Pas de priorisation des développements.	**Réduire le risque d'échec du projet** **Apporter de la qualité logicielle**
Dépassements des délais.	**Réduire le risque d'échec du projet**
Pas de *tests unitaires*, constatation tardive de bogues, régression.	**Apporter de la qualité logicielle**
Tests fonctionnels réalisés manuellement et incomplets. Constatation tardive de bogues.	**Apporter de la qualité logicielle**
Les retours clients ne sont pas toujours formalisés et relayés entre les gérants de l'entreprise. Perte d'information.	**Avoir plus de visibilité sur le projet**
Effet tunnel sur certains projets.	**Réduire le risque d'échec du projet** **Apporter de la qualité logicielle**
Le planning prévisionnel n'est plus mis à jour une fois le projet en phase de pilotage.	**Avoir plus de visibilité sur le projet**
Absence de reporting de pilotage permettant de mesurer l'écart entre le réalisé et l'attendu.	**Avoir plus de visibilité sur le projet**
Pas de validation des fonctionnalités développées. Demandes incessantes du client après le déploiement du produit.	**Avoir plus de visibilité sur le projet**

On peut dégager **quatre grands besoins** pour l'entreprise :

Bénéficier d'une plus grande adaptabilité avec le client

L'équipe doit être en mesure de recueillir et de traiter les retours du client afin de proposer un produit plus adapté à la demande.

Avoir une meilleure visibilité sur le projet

Aussi bien pour le client que pour l'entreprise, celle-ci doit être en mesure d'évaluer de façon plus précise l'état d'avancement du projet et de mesurer l'écart entre le réalisé et l'attendu par le client.

Apporter de la qualité logicielle

Nous entendons par qualité logicielle :

- La conformité du produit réalisé aux exigences du client.
- La fiabilité du produit grâce à des tests.

- La maintenabilité du produit.

Réduire le risque d'échec du projet

Nous avons constaté différents dysfonctionnements pouvant mener à l'échec du projet. L'entreprise doit mettre en œuvre des méthodes ou des outils afin de réduire ce risque d'échec.

5. Résumé

Dans ce chapitre, nous avons défini ce qu'était un projet et avons recensé et classé les dysfonctionnements existants en matière de gestion de projet au sein de la société CosiWeb.
Nous nous sommes ensuite appuyé sur la méthode Méhari, habituellement employée dans la sécurité informatique, afin d'évaluer quels sont les dysfonctionnements actuels qui font courir le plus grand risque d'échec des projets menés et avons fait émerger les principaux besoins de CosiWeb en la matière.
Nous allons maintenant dresser un état de l'art des méthodes classiques et agiles de gestion de projet et tenterons de comparer les deux approches ; nous finirons par voir si l'agilité est une réponse pertinente aux besoins que nous venons de repérer.

Les méthodes de gestion de projet

Dans la suite de ce mémoire, nous ferons une distinction nette entre les méthodes de gestion de projet classiques et les méthodes agiles.

Les méthodes de gestion de projet classiques se basent sur une approche dite « *prédictive* » ou « *déterministe* ». Toutes les étapes du projet doivent être planifiées dès son lancement et strictement respectées.

Les méthodes classiques reposent sur le fait que les étapes du projet se suivent de façon séquentielle et qu'une étape ne peut être entamée sans que la précédente ne soit terminée. Par ailleurs, une modification du projet en aval aura des conséquences potentiellement importantes en amont. C'est pour cela qu'il existe une grande rigidité dans l'adoption de ces méthodes car un changement inattendu dans une étape du projet peut avoir pour conséquence des pertes financières voire un échec du projet.

Le développement informatique s'est basé depuis les années soixante-dix et se base encore sur ces méthodes classiques. Hors l'entreprise est confrontée à des demandes de clients qui évoluent fréquemment durant le projet. Suite à ce constat, des méthodes plus souples, dites « *agiles* », sont apparues au début des années 2000, afin d'adapter le projet à des demandes qui peuvent évoluer. Plutôt que d'essayer de prédire toutes les étapes du projet, le principe est de s'adapter à la demande avec une grande réactivité.

Nous ne chercherons pas dans ce mémoire à balayer d'un revers de la main les méthodes classiques ; celles-ci proposent par exemple des pratiques qui sont reprises dans certaines méthodes agiles, comme les tests unitaires et fonctionnels. L'objet de cette section est d'apprécier avec objectivité les apports que peuvent fournir les méthodes agiles en les comparant notamment avec celles dites classiques.

1. Les méthodes classiques

1.1. Fondements théoriques

Les méthodes classiques existent depuis plusieurs décennies. En effet, le cycle de vie dit « *en cascade* » a été formalisé par Royce (1970), un informaticien américain. Les méthodes classiques ont été conçues pour apporter un cadre formel et structuré dans la gestion de projet.

1.2. Les principales méthodes

1.2.1. Le cycle « en cascade »

Le cycle en cascade repose sur les principes suivants (Messager & Tabaka, 2010) :

- Le cycle de vie du projet est constitué de phases séquentielles.
- On ne peut passer à la phase suivante que lorsque les livrables de la phase actuelle sont validés.
- On ne peut pas revenir à la phase précédente une fois celle-ci validée.
- Tous les besoins du projet sont exprimés et recueillis lors de la première phase.
- Le système est conceptualisé et validé avant la phase de développement.
- Les tests techniques et fonctionnels ne démarrent qu'après la phase de développements.
- L'intégration et la mise en production ont lieux après correction des anomalies.

Figure 6 : Phases du cycle en « cascade ».

On peut se rendre compte de la rigidité de cette approche. Cela peut être un atout dans un contexte dans lequel les besoins ne vont pas évoluer durant la vie du projet. Mais cela peut être un frein en matière de développement informatique. Par ailleurs, une anomalie dans la conception peut être décelée tardivement et entrainer un retard et dépassement du budget considérable.

Pour l'anecdote, Royce pointait déjà du doigt en 1970 les limites de cette approche et préconisait une approche itérative pour y pallier (Royce, 1970). Le fondateur du cycle de développement le plus classique était déjà un agiliste !

Une variante du cycle en « *cascade* » est ainsi apparue afin d'identifier plus rapidement les anomalies en limitant le retour aux étapes précédentes. Il s'agit du cycle en « *V* ».

1.2.2. Le cycle en « V »

Le modèle du cycle en « *V* » a été élaboré pour éviter les problèmes de réactivité du cycle en cascade en cas d'anomalies.

Le modèle est constitué d'une branche descendante qui comprend les étapes de conception du projet et d'une branche ascendante qui contient les étapes de tests. La jonction des deux branches constitue la phase de codage du produit.

Chaque étape d'une branche est en vis-à-vis avec l'étape de la branche opposée, les étapes de la branche montante doivent renvoyer de l'information à leur vis-à-vis lorsqu'une anomalie est détectée. Cela permet de déceler plus rapidement les problèmes que dans un cycle en cascade. Par ailleurs, cela offre la possibilité de préparer dans les étapes descendantes les besoins des étapes montantes. Ainsi les résultats attendus des tests unitaires seront définis dans la conception détaillée.

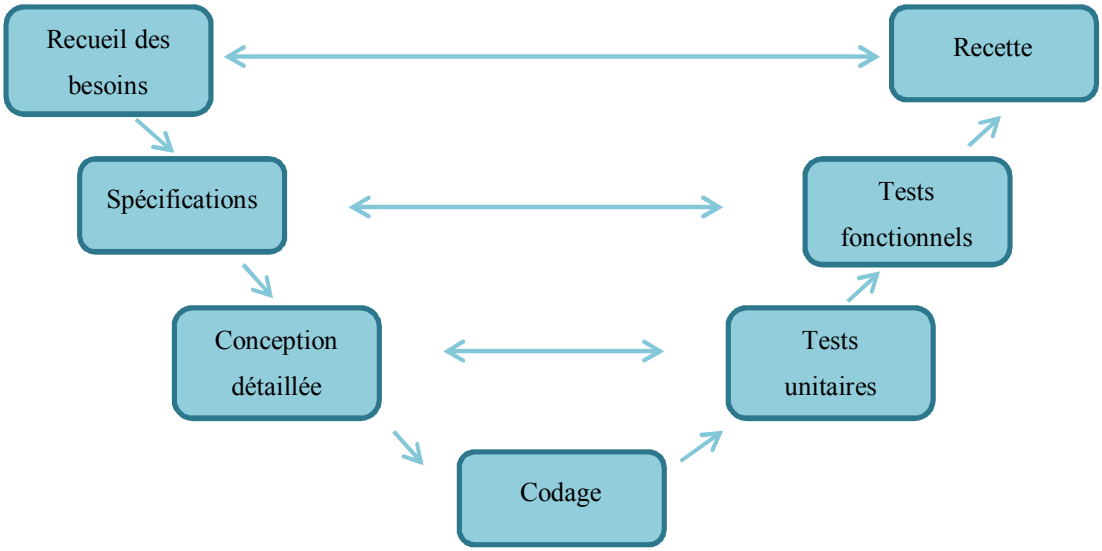


Figure 7 : Phases du cycle en « V ».

1.3. Critiques des méthodes classiques

Nous l'avons vu précédemment, une des critiques faites aux méthodes classiques est leur approche dite « *prédictive* » qui apporte beaucoup de rigidité dans la gestion des projets. D'autres écueils sont pointés du doigt par Boisvert (2011), Fernandez (2013) et Messager (2010), nous allons lister les principaux.

1.3.1. Le risque de l'effet tunnel

C'est certainement un des principaux problèmes des méthodes classiques.

Prenons l'exemple d'un projet qui dure six mois : l'entreprise fait un recueil des besoins auprès du client durant un mois, puis se lance dans la phase de développement durant quatre mois. Le client ne voit le résultat que le sixième mois… entre temps, les besoins de ce dernier ont pu changer et des fonctionnalités sont devenues obsolètes ou inadaptées.

1.3.2. L'anticipation et la détection des risques

Comme nous venons de le voir dans les cycles en « *cascade* », les tests (unitaires et fonctionnels) sont réalisés après la phase de codage. Toutefois, l'impact d'une anomalie va augmenter avec l'avancement du projet. Les correctifs seront plus longs à implémenter et il sera nécessaire de s'assurer de la non régression du code.

1.3.3. La résistance au changement

Dans les méthodes classiques, les fonctionnalités à implémenter sont définies au lancement du projet, voire dans la réponse de l'entreprise dans le cas d'appel d'offres. Ces fonctionnalités sont détaillées dans le contrat initial entre le client et l'entreprise et toute modification de ces dernières est un sujet sensible car pouvant conduire à des dépassements de délais et de coûts. Dans le meilleur des cas, l'entreprise peut négocier l'amélioration d'une fonctionnalité contre l'abandon d'une autre… dans le pire des cas, le produit livré peut ne plus correspondre aux besoins du client. L'entreprise a donc tendance à refuser tous changements dans les demandes de modifications des fonctionnalités initiales.

Nous allons maintenant présenter l'alternative offerte par les méthodes agiles et comprendre en quoi elles peuvent pallier les manques des méthodes classiques, que nous venons d'aborder.

2. Les méthodes agiles

2.1. Fondements théoriques

Avant de dresser un bref historique de l'apparition du mouvement agile, nous allons expliquer les principes fondamentaux de l'approche, puis nous présenterons les avantages et les bienfaits théoriques des méthodes agiles.

2.1.1. Définition d'une méthode agile

« *Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif, avec juste ce qu'il faut de formalisme. Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients.* » (Messager & Tabaka, 2010)
Deuff (2013) écrit que les méthodes agiles sont « *des modèles itératifs et incrémentaux qui visent à répondre au mieux aux besoins exprimés par leurs commanditaires, en offrant une grande réactivité par rapport à leurs demandes* » (Deuff & Cosquer, 2013).

Nous retrouvons dans ces deux définitions les termes « *itératif* » et « *incrémental* ». Dans un développement itératif, le projet est découpé en plusieurs étapes d'une durée identique (en général quelques semaines), nommées « *itérations* ». A chaque fin d'itération, une version fonctionnelle du produit est attendue par le client pour validation, nous parlons d'une version intermédiaire du produit final et non d'un prototype. Les fonctionnalités sont ainsi progressivement intégrées au produit qui est conçu de façon « *incrémentale* » (Messager & Tabaka, 2010).

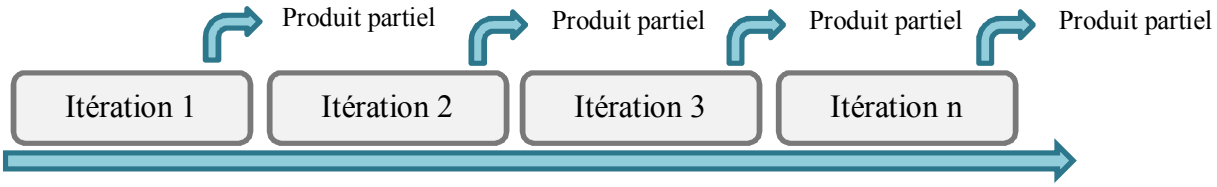


Figure 8 : Représentation d'un développement itératif et incrémental.

Les deux définitions mettent aussi l'accent sur la prise en compte de l'évolution des besoins du client. En effet, un des aspects fondamentaux des méthodes agiles est de partir du constat que les demandes des clients peuvent évoluer dans le cycle de vie du projet et qu'il est nécessaire de les prendre en compte pour livrer au final le produit le plus adapté possible à leur besoins.

Il est important de faire une distinction entre « *méthode* » et « *pratique agile* ». Une entreprise peut en effet décider de suivre certaines pratiques d'une méthode, sans adopter la méthode dans son ensemble. Elle peut aussi décider de mettre en œuvre des pratiques de plusieurs méthodes agiles, comme le souligne Boisvert (2011).

Les méthodes agiles ont pour point commun de partager les valeurs et les principes consignés dans un document : le « *manifeste agile* » (confère annexes page 102).

2.1.2. Qu'est-ce que l'agilité ? Valeurs et principes

Le mouvement agile voit son origine aux États-Unis, en février 2001, suite au constat du taux d'échec de plus en plus important des méthodes de gestion de projet classiques dans les années 1990 (Messager & Tabaka, 2010). Dix-sept experts[9] du développement logiciel ayant pour la

[9] Parmi ces spécialistes, nous pouvons citer Kent Beck, qui a notamment popularisé le développement piloté par les tests (Test Driven Developpement ou TDD) ou le concept de design simple du code source, Ward Cunningham, l'inventeur du concept de wiki, ou encore Martin Fowler et Alistair Cockburn, des pionniers de l'agilité.

plupart mis au point de nouvelles méthodes se réunissent[10] afin de débattre de nouvelles pratiques et de mettre en avant des valeurs communes (Fernandez et al., 2013).

Du résultat de cette réflexion est né le terme « *agile* », ainsi que quatre valeurs inscrites dans le manifeste pour le développement agile de logiciels (Beck et al., 2001) :

> *« **Les individus et leurs interactions** plus que les processus et les outils*
>
> ***Des logiciels opérationnels** plus qu'une documentation exhaustive*
>
> ***La collaboration avec les clients** plus que la négociation contractuelle*
>
> ***L'adaptation au changement** plus que le suivi d'un plan*
>
> *Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers. »*

De ces quatre valeurs découlent douze principes sous-jacents (Beck et al., 2001) (confère annexes page 102).

Il est important de préciser que les deux méthodes agiles les plus utilisées ne sont pas apparues suite à cet événement. Les fondements de « *Scrum* » étaient présentés en 1995 lors de la conférence OOPSLA (Schwaber, 1995) et « *Extreme Programming* » a été lancé officiellement en 1999.

2.1.3. Avantages escomptés de l'agilité

Boisvert (2011) livre plusieurs avantages à l'adoption de l'agilité dans un projet :

- Fournir rapidement de la valeur au client : la valeur est apportée par les fonctionnalités validées par le client.
- Améliorer la qualité du produit en réduisant les défauts de conception (par les tests unitaires notamment).
- Réduire les délais de mise en production par des livraisons fréquentes du produit.
- Respecter le budget par la détection des risques le plus tôt possible et par la possibilité pour l'équipe de surveiller sa vélocité dans la réalisation des tâches.
- Améliorer la collaboration et la communication entre le client et l'entreprise.

---

[10] Ces différents protagonistes se réunirent dans le cadre informel d'une station de ski pour tenter de trouver un courant de pensée commun et pour… manger et se relaxer ! Alistair Cockburn n'était d'ailleurs pas convaincu qu'il puisse émerger d'idées communes avec autant de participants et fut d'autant plus surpris et enthousiaste à l'issue de la rencontre (Highsmith, 2001).

2.2. Les deux principales méthodes

Une enquête nationale a été menée par Legardeur et Sutherland (2009) afin de connaitre les pratiques d'entreprises de tailles variées en matière de gestion de projet agile. Il en est notamment ressorti que les deux méthodes agiles les plus employées étaient Scrum et eXtreme Programming (XP). En troisième position arrive la méthode Lean. Cette dernière méthode, héritée du système de production de Toyota, n'est pas à proprement parler une méthode agile selon Messager (2010), même si certaines pratiques peuvent être employées dans le cadre d'une démarche agile. Nous centrerons notre présentation sur les deux principales méthodes agiles pour deux raisons :

- Ces deux méthodes sont les plus documentées et leur audience est la plus forte auprès de la communauté agile. Nous avons d'ailleurs pu nous en apercevoir lors de « *l'agile tour Toulouse* »[11].
- Nous le verrons dans la suite de ce mémoire, Scrum et XP sont deux méthodes très complémentaires, la première dans sa démarche de gestion de projet et la seconde pour ses pratiques d'ingénierie logicielle.

Nous allons présenter chacune de ces deux méthodes selon leurs principes, le rôle des acteurs, les outils employés et leur mode opératoire. Nous n'avons pas pour objectif d'en faire une présentation exhaustive mais plutôt d'en fournir les notions essentielles. Nous nous appuierons sur les ouvrages de Fernandez (2013) et Messager (2010) pour les détailler.

2.2.1. La méthode Scrum

La méthode Scrum a été décrite pour la première fois dans un ouvrage par Schwaber et Beedle en (2001).

2.2.1.1. Valeurs

Scrum est une méthode qui se focalise sur la gestion de projet, elle ne couvre aucune pratique d'ingénierie logicielle. Elle repose sur trois valeurs :
La transparence : l'avancement du projet doit être visible des membres de l'équipe en charge et du propriétaire du produit.

[11] Evénement annuel autour de l'agilité. http://tour.agiletoulouse.fr/

Le contrôle : on doit pouvoir vérifier fréquemment les écarts de délais et de réalisation des tâches.

L'adaptation : on doit pouvoir faire des ajustements lorsque des écarts sont constatés afin de toujours répondre aux besoins du propriétaire du produit.

2.2.1.2. Rôles

La responsabilité de la conduite du projet est répartie sur trois rôles :

Le ***product-owner*** (propriétaire du produit ou PO) qui représente le client ou les utilisateurs finaux. Il va communiquer les besoins et la vision du produit à l'équipe de développement. Il sera impliqué à des moments clefs durant le développement.

L'***équipe Scrum*** en charge du développement du produit. Elle est pluridisciplinaire et n'est pas hiérarchisée, les décisions sont prises de façon collégiale.

Le ***Scrum-master*** (animateur d'équipe) fait la liaison entre le *product-owner* et l'*équipe Scrum*, il peut faire partie de l'équipe et est en charge de la bonne mise en pratique de la méthode.

2.2.1.3. Outils

Il existe cinq principaux outils :

- Le ***product-backlog*** (carnet de produit) permet de lister les exigences données par le *product-owner* et classées par priorités. Le *product-backlog* peut être modifié en fonction de l'évolution des besoins du *product-owner*. Ces exigences sont appelées des *user stories* (récits d'utilisateur ou scénarios) et décrivent une fonctionnalité à développer. Concrètement, les *user stories* sont des phrases simples qui répondent à trois interrogations : qui ? Quoi ? Pourquoi ? Voici un exemple :
  « *En tant qu'utilisateur, je veux pouvoir rechercher un étudiant par son numéro afin de le retrouver rapidement.* »
- Les *user stories* sont ensuite traduites et répertoriées dans le ***sprint-backlog*** (carnet de l'itération) sous forme de tâches à réaliser au cours d'une itération.
- Un graphique de progression, le ***Burndown chart***, permet de visualiser l'avancement des tâches dans le temps. Il peut être complété par un graphe présentant la valeur acquise, c'est-à-dire les fonctionnalités validées par le client et qui lui sont immédiatement utiles.
- Le ***task board*** se présente sous la forme d'un tableau sur lequel on liste dans une première colonne les *user stories* à l'aide de post-it, puis les fonctionnalités du *sprint-*

backlog dans quatre colonnes pour représenter leur avancement : à faire, en cours, à valider, fait.

- Enfin, la ***roadmap*** permet d'identifier les moments importants du projet, elle est mise à jour de façon régulière.

2.2.1.4. Pratiques

Un projet sous Scrum est découpé en itérations de quelques semaines (un mois en général) appelées *Sprint*.

Au lancement du projet, les *user stories* du *product-owner* sont consignées dans le *product-backlog*. Avant chaque *Sprint* durant le *sprint planning meeting*, on sélectionne les *user stories* prioritaires qui seront développées et livrées en fin de *Sprint*.

Au cours du sprint, une réunion d'avancement a lieu de façon quotidienne avec les membres de l'équipe (le *daily Scrum meeting* ou *mêlée quotidienne*), afin de vérifier les écarts aux objectifs du *Sprint*.

A la fin du *Sprint*, une version partielle mais opérationnelle du produit est présentée au *product-owner* durant le *sprint review meeting* ou *revue de sprint*. Le *product-owner* est censé pouvoir arrêter le projet s'il juge le produit répondant à ses besoins, dans le cas contraire, un nouveau *Sprint* est commandé.

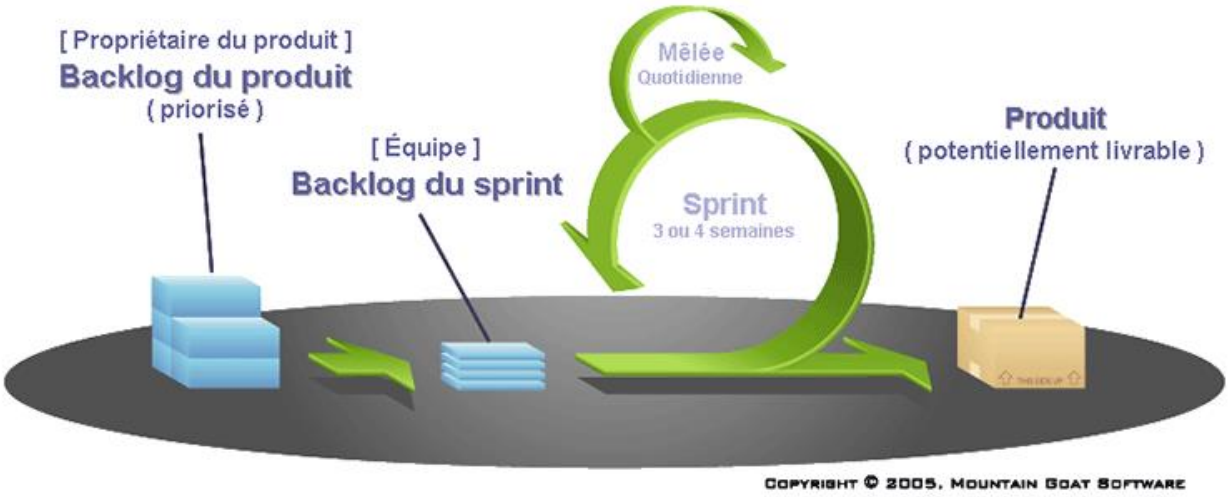


Figure 9 : Le cycle de vie Scrum ("Scrum Overview for Agile Software Development," 2005).

2.2.2. La méthode eXtreme Programming (XP)

La méthode XP a été décrite dans l'ouvrage de Beck (2000).

2.2.2.1. Valeurs

XP est une méthode agile focalisée sur des pratiques d'ingénierie logicielle comme les tests unitaires et fonctionnels, voire le développement piloté par les tests unitaires (TDD), mais aussi la programmation en binôme ou encore l'intégration continue. Elle repose sur quatre valeurs :

La communication : il est indispensable que les intervenants d'un projet communiquent afin d'éviter des situations de blocage et la découverte tardive de problèmes.

La simplicité : la solution la plus simple est la meilleure. La simplicité doit se retrouver aussi bien dans les choix techniques que dans les besoins du client ou les processus à développer.

Le feedback : le retour d'information est indispensable d'un point de vue technique avec les tests unitaires ou fonctionnels pour valider le fonctionnement du produit et d'un point de vue organisationnel avec des livraisons fréquentes du produit pour validations du client.

Le courage : il en faut pour reprendre un code existant, faire de la programmation en binôme…

2.2.2.2. Rôles

Dans la littérature, à notre connaissance, les rôles de la méthode XP sont peu abordés, hormis dans l'ouvrage de Fernandez (2013) qui en distingue huit : le programmeur, le designer, le testeur, le tracker, le coach, le manager de projet, le client et le chef de produit. Par ailleurs, au sein de la communauté agile que nous avons pu côtoyer dans le cadre de ce mémoire, ce sont les pratiques de la méthode XP qui sont mises en avant plus que les rôles ou les outils.

En pratique, certains rôles sont confondus ; ainsi, le programmeur est souvent aussi le testeur et le client est généralement le chef de produit. Il faut retenir que la méthode XP fait la part belle aux activités de programmation plus qu'à la gestion de projet, qui s'avère moins formalisée qu'avec la méthode Scrum.

2.2.2.3. Outils

Il n'existe en fait que deux outils : les ***story-cards*** qui sont des cartes sur lesquelles on recense les *user stories* du client ; et le ***story-board*** visible en permanence par l'équipe, sur lequel on affiche les story-cards. On précise dans chaque *story-card* les ressources nécessaires pour l'accomplissement de ces *user stories* ainsi que les tests associés. Ces *story-cards* peuvent être complétées par des post-it pour visualiser l'avancement du projet, à la manière du *task board* de la méthode Scrum.

2.2.2.4. Pratiques

Contrairement à la méthode Scrum, la méthode XP ne se focalise pas seulement sur la gestion de projet mais propose des pratiques et des outils très poussés pour améliorer la collaboration entre les développeurs et optimiser la programmation. Il est à noter qu'XP offre une description des processus de gestion de projet moins détaillée que celle de Scrum.

XP fonctionne comme Scrum, selon le principe de cycles rapides de développement, les itérations. Un projet XP est composé d'itérations qui s'articulent selon les étapes suivantes :

- Au lancement d'une itération, le client et l'équipe de développement consignent dans des *story-cards* les *user stories* à développer, cela se déroule lors d'un ***planning poker***.
- Chacune des *user stories* est transformée en tâche qui seront réalisées durant l'itération selon les pratiques collaboratives et de programmation que nous allons aborder en suivant (TDD, intégration continue…).
- A la fin de l'itération, si les *user stories* réalisées sont validées et que le client fournit de nouveaux scénarios à développer, une nouvelle itération démarre. Il est à noter que le client possède une forte implication dans le projet et est idéalement présent sur le site pour les réunions de *planning poker* notamment.

Les principales **pratiques collaboratives** de la méthode XP sont :

- Le ***pair programming*** (programmation en binôme) dont le principe est de faire travailler durant un laps de temps assez court (une heure par exemple) deux développeurs sur une machine afin de produire un code de plus grande qualité. Cette pratique n'est possible qu'en respectant un ***coding standard*** (règles de codage) commun pour l'équipe.
- L'**intégration continue** qui consiste à intégrer au produit les modifications faites par les développeurs de façon continue (au moins quotidiennement) afin de détecter les problèmes au plus tôt grâce à des **tests d'intégration**.

Les **pratiques relatives à la programmation** sont les suivantes.

Le ***simple design*** (conception simple) proposé par Beck (2000) consiste en quatre règles qu'un programmeur doit respecter en écrivant son code :

- Passer tous les tests.
- Ne pas dupliquer le code.
- Révéler ses intentions par le code.
- Réduire le nombre de classes et de méthodes.

Le code source doit être soumis à des **tests unitaires** et la méthode propose la pratique du **Test Driven Development (TDD)**. Le TDD consiste à écrire le test unitaire avant le code. Il se déroule suivant un cycle en cinq étapes :

- Ecrire un premier test.
- Lancer le test qui doit échouer pour vérifier qu'il est valide (en effet le code n'existe pas encore).
- Ecrire le code permettant de faire passer le test.
- Vérifier que le test passe.
- **Refactoriser le code**, c'est-à-dire l'améliorer en suivant par exemple les règles du *simple design*.

Les tests unitaires pourront être suivis par des **tests fonctionnels** afin de valider les fonctionnalités développées.

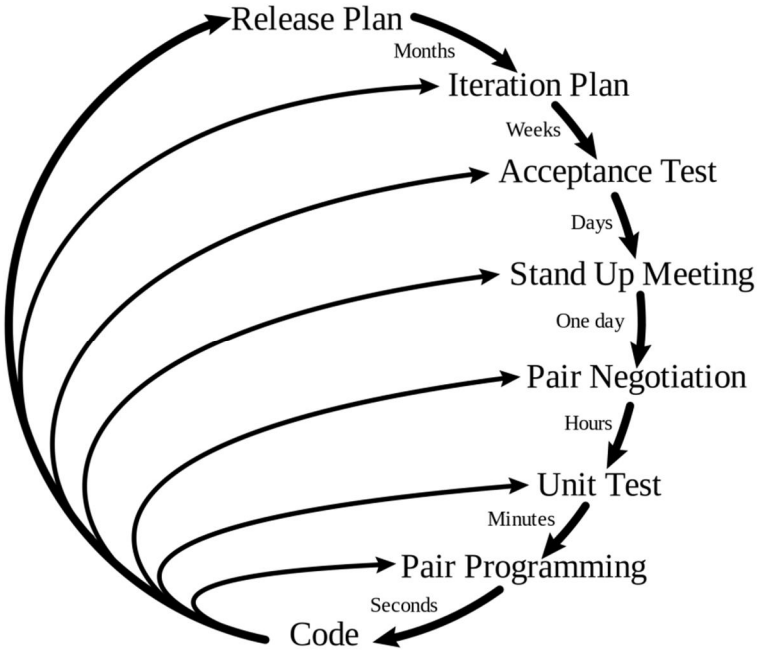


Figure 10 : Le cycle de vie eXtreme programming (XP) (Valentine, 2011).

2.3. Comparaison des méthodes Scrum et eXtreme Programming (XP)

En nous appuyant sur les ouvrages de Fernandez (2013) et de Messager (2010), nous pouvons mettre en avant les forces et les faiblesses des deux méthodes.

Méthode	Points forts	Points faibles
Scrum	- Se focalise sur la création de valeur ajoutée pour le *product-owner*. - Favorise la communication et la collaboration au sein de l'équipe et avec le *product-owner*. - Propose un cadre de travail qui peut être utilisé sur des projets autres qu'informatiques. - Propose un programme de certification. - Ne nécessite pas d'outillage complexe pour sa mise en place. - Méthode agile la plus répandue en France.	- N'aborde pas de pratiques d'ingénierie logicielle. - Changements limités avec des itérations de trente jours et des tâches à réaliser figées durant celles-ci.
eXtreme Programming (XP)	- Solides pratiques d'ingénierie logicielle. - Les tests présents à tous les niveaux du développement, favorisent l'émergence de qualité du code. - Des itérations courtes et une implication importante du client, donc un feedback fréquent.	- Demande de l'expérience et du temps pour développer certaines pratiques techniques : TDD, intégration continue. - Le *pair programming* n'est pas toujours apprécié des développeurs. - Nécessite beaucoup de rigueur, notamment dans la conduite des tests. - Faible documentation, méthode limitée dans le cas d'un projet évolutif ou multi équipes.

Tableau VI : Forces et faiblesses des méthodes Scrum et eXtreme Programming (XP).

Nous pouvons voir que certains points faibles d'une méthode sont les points forts de l'autre. De là à dire que ces méthodes peuvent être complémentaires, il n'y a qu'un pas…

2.4. Vers une hybridation des deux méthodes

Nous l'avons vu, Scrum et XP proposent chacune leur façon d'aider l'équipe. Alors que Scrum propose des activités permettant d'améliorer la gestion de projet, XP propose des outils et des pratiques d'ingénierie logicielle.

Nous retenons de la comparaison entre ces deux méthodes leur grande complémentarité et les avantages que nous pouvons escompter en combinant leurs pratiques. Selon Boisvert (2011), la combinaison des méthodes agiles est très répandue, leurs pratiques sont compatibles comme dans

le cas de Scrum et XP et permettent en se complétant d'obtenir une méthode agile hybride adaptée à une organisation ou un projet.

3. Comparaison entre approche classique et approche agile

Nous venons de voir que les méthodes agiles offrent une approche bien différente de celle proposée dans les méthodes classiques. Le tableau ci-dessous s'appuie sur l'ouvrage de Messager (2010) et nous permet de faire une synthèse comparative des deux approches.

Thème	Approche classique	Approche agile
Cycle de vie	Séquentiel, en cascade ou en V sans rétroaction possible.	Itératif et incrémental.
Planification	Prédictive, les besoins sont définis dès le lancement du projet et les développements planifiés sans modification possible.	Adaptative, des ajustements peuvent être effectués à chaque fin d'itérations.
Changement	Résistance au changement.	Intégré dans l'approche.
Equipe	Une équipe avec des ressources spécialisées. Dirigée par le chef de projet.	Une équipe responsabilisée où l'initiative et la communication sont encouragés. Soutenue par le chef de projet.
Qualité	Contrôle qualité tardif, à la fin du cycle de développement. Le client découvre le produit fini.	Contrôle qualité régulier. Le client visualise les résultats tôt et fréquemment.
Suivi de l'avancement	Mesure de la conformité aux plans initiaux. Analyse des écarts à posteriori.	Deux indicateurs d'avancement : le travail restant à faire et la valeur acquise par le client (fonctionnalités utiles au client, implémentées et fonctionnelles).
Documentation	Produite en quantité importante comme support de communication, de validation et de contractualisation.	Réduite au strict nécessaire au profit d'incréments opérationnels du produit et d'échanges fréquents avec le client.
Gestion des risques	Processus distinct de gestion des risques.	Intégré dans le processus global, avec responsabilisation de chacun (équipe et client) dans l'identification et la résolution des risques.
Mesure du succès	Respect des engagements initiaux en termes de coûts, de budget et de qualité.	Satisfaction du client par la livraison de valeur ajoutée.

Tableau VII : Synthèse comparative entre approche classique et approche agile.

Suite à ce que nous venons de voir, l'agilité semble être une alternative séduisante, voire incontournable, aux approches classiques. Examinons pour terminer si les besoins de CosiWeb que nous avons mis en avant précédemment peuvent être comblés en théorie par l'emploi de l'agilité.

4. Mise en relief des besoins de CosiWeb et des apports escomptés de l'agilité

Comme nous venons de le voir, la méthode XP est portée sur les techniques d'ingénierie logicielle et Scrum vise à améliorer la gestion de projet. Nous reprenons dans le tableau suivant les dysfonctionnements constatés au sein de CosiWeb (voir page 35) et faisons correspondre les méthodes agiles qui nous semblent les plus adaptées pour les pallier.

Etape du projet	Dysfonctionnement constaté	Méthode agile préconisée
Formulation des besoins du client	Pas de prise en compte de tous les besoins du client.	**Scrum**
Recueil des besoins du client	Mauvaise compréhension des besoins du client.	**Scrum**
Planification	Pas de priorisation des développements.	**Scrum**
Planification	Dépassements des délais.	**Scrum**
Conception du produit	Pas de *tests unitaires*, constatation tardive de bogues, régression.	**XP**
Conception du produit	*Tests fonctionnels* réalisés manuellement et incomplets. Constatation tardive de bogues.	**XP**
Communication au sein de l'entreprise	Les retours clients ne sont pas toujours formalisés et relayés entre les gérants de l'entreprise. Perte d'information.	**Scrum**
Communication avec le client	Effet tunnel sur certains projets.	**Scrum**
Suivi du projet	Le planning prévisionnel n'est plus mis à jour une fois le projet en phase de pilotage.	**Scrum**
Suivi du projet	Absence de reporting de pilotage permettant de mesurer l'écart entre le réalisé et l'attendu.	**Scrum**
Validation des travaux	Pas de validation des fonctionnalités développées. Demandes incessantes du client après le déploiement du produit.	**Scrum**

Tableau VIII : Besoins de CosiWeb et méthodes agiles pouvant y répondre.

5. Résumé

Dans ce chapitre, nous avons fait un état de l'art des méthodes classiques et agiles en matière de gestion de projet et les avons comparées selon différents critères. Pour finir, nous avons repris les

besoins que nous avions mis en relief dans le chapitre précédent et avons émis l'hypothèse que les méthodes agiles Scrum et XP sont en mesure de les combler. Nous allons maintenant décrire la mise en pratique de l'agilité au sein de CosiWeb et voir si elle a permis d'apporter des solutions. Nous nous appuierons sur l'expérimentation que nous avons pu en faire sur deux projets candidats et présenterons l'application ASIPAT que nous avons développée afin de pallier notamment les manques en matière de tests fonctionnels.

Mise en pratique de l'agilité

Nous avons volontairement choisi le titre de « *mise en pratique de l'agilité* » plutôt que celui de « *mise en pratique des méthodes agiles* ». En effet, nous n'avons pas mis en œuvre une méthode dans sa globalité mais plutôt des pratiques agiles : nous le verrons plus en détail dans la section concernant l'analyse des limites de l'agilité en page 93, l'un des principaux obstacles de la mise en œuvre d'une méthode agile dans son ensemble est la contractualisation spécifique et complexe qu'elle représente. Cela ne diminue en rien les apports dont nous avons pu bénéficier en mettant en œuvre ces pratiques dans deux projets candidats que nous allons présenter.

1. Présentation des deux projets candidats : CAPTOR et CRCT-INSERM

Nous n'allons pas décrire le déroulement de ces projets de façon séquentielle et exhaustive mais nous préférons mettre l'accent sur les pratiques expérimentées, les outils utilisés et les bénéfices obtenus.

1.1. Présentation des projets

Les deux projets se sont déroulés sur des périodes de onze semaines.

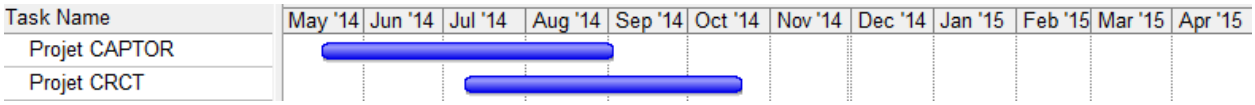


Figure 11 : Déroulement dans le temps des projets candidats à l'agilité.

CAPTOR (Cancer Pharmacology of Toulouse-Oncopole and Region) est un vaste programme de recherche de l'Oncopole de Toulouse consacrés à la recherche sur les médicaments anti-cancéreux et réunissant l'Université Toulouse III, le Centre Hospitalier Universitaire de Toulouse, l'institut Claudius Regaud et le Centre de Recherche en Cancérologie de Toulouse (CRCT). Nous devions concevoir le site web du projet CAPTOR[12].

Le CRCT est un centre de recherche contre le cancer, situé sur l'Oncopole de Toulouse et réunissant dix-huit équipes de recherche. Nous avons opéré la refonte du site web du CRCT[13].

Notre mission durant ces deux projets a été la réalisation de sites web portails présentant ces entités à des publics variés : entreprises, étudiants, chercheurs, journalistes et grand public.

---

[12] Voici l'adresse du site réalisé : http://www.captor-cancer.fr

[13] Voici l'adresse du site réalisé : http://www.crct-inserm.fr

1.2. Critères de choix de ces projets

Nous nous sommes inspirés des critères proposés par Boisvert (2011) pour sélectionner et soumettre ces projets à l'agilité.

Durée du projet : le projet ne doit pas être trop court pour observer les apports mais aussi les biais de la mise en pratique de l'agilité. Un projet de quelques mois est idéal.

Appui du client : le client doit comprendre les enjeux d'une démarche agile et accepter d'adapter son fonctionnement en étant notamment très impliqué dans la vie du projet.

Une taille d'équipe raisonnable : pour lancer une démarche agile, Boisvert (2011) conseille de ne pas dépasser une équipe de développement de dix membres. Les équipes que nous avons montées durant ces projets n'ont jamais dépassé quatre membres.

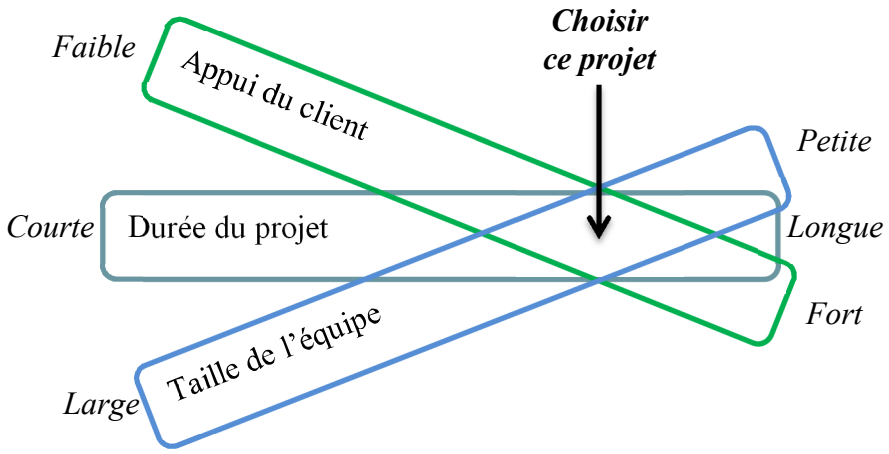


Figure 12 : Les critères pour choisir un projet candidat au démarrage de l'adoption de l'agilité (Boisvert & Trudel, 2011).

Les deux projets ont répondu parfaitement à ces critères avec une durée s'étalant sur onze semaines, de par la volonté affirmée de nos commanditaires à expérimenter ces pratiques et enfin par la taille de notre équipe.

Pour revenir sur l'adhésion de nos clients à une démarche agile, nous nous sommes questionnés lors du premier projet (CAPTOR) sur leur réaction lors de la présentation de l'agilité, quelques jours avant la réunion de lancement. Il se trouve que ces derniers avaient déjà entendu parler de ce courant et étaient très curieux de découvrir de quoi il était question. Nous avons aussi ressenti cet engouement avec nos interlocuteurs sur le projet suivant. Nous pensons toutefois qu'il ne faut pas généraliser car nous pouvons relater la réaction d'un client du secteur privé qui nous a dit lorsque nous lui avons présenté le déroulement d'un projet Scrum : « *Quand je vois à quel point*

la façon dont vous comptez gérer ce projet est structurée, je me dis que c'est un travail plus conséquent que je ne le pensais et que cela va me coûter cher !».

Pour finir, je citerai Sébastien Balard, Scrum-master chez Prometil[14] et avec qui je me suis entretenu : *« tous les projets que j'ai menés dans l'agilité et dans lesquels le product owner* (propriétaire et utilisateur final du produit, abréviation PO) *était aussi le payeur, ont été un échec…* ». Nous avons donc décidé pour ces projets candidats de rajouter un dernier critère : **choisir un projet dans lesquels les PO ne sont pas les payeurs**.

1.3. Choisir la méthode agile la plus adaptée à ces projets

Après avoir choisi les premiers projets que nous mènerions avec des pratiques agiles, nous avons décidé d'employer la méthode Scrum pour les raisons suivantes :

- L'essentiel des travaux des deux projets concernaient de la conception graphique, de la rédaction et de l'intégration de contenus, de la traduction et de la réalisation de médias audiovisuels. Il n'y a pas eu de développements techniques particuliers nécessitant des pratiques agiles d'ingénierie logicielle proposées par XP.
- Pour reprendre Fernandez (2013) et Messager (2010), la méthode Scrum ne nécessite pas d'outillage complexe à mettre en œuvre…
- … et c'est aussi la méthode agile la plus répandue en France. Ainsi, lorsque nous présentons Scrum à nos interlocuteurs, ces derniers en ont souvent entendu parler et sont plutôt curieux de pratiquer.

Bien que nous ayons employé uniquement Scrum durant ces projets, nous l'avons vu précédemment, l'idéal est de combiner Scrum avec XP, pour les apports de ce dernier dans le développement technique.

2. Déroulement d'un projet Scrum

Nous avons mené les deux projets pilotes selon une démarche similaire. Nous allons présenter la façon dont nous nous sommes appropriés Scrum. En effet, pour en avoir discuté avec différents Scrum-master, la méthode fournit un cadre méthodologique que l'on peut et que l'on doit adapter à sa structure et à ses projets.

---

[14] http://www.prometil.com/

2.1. Rôles, outils et pratiques employées

Nous avons employé les rôles, les outils et les pratiques de la méthode Scrum que nous avons présentés en page 44.
J'ai joué le rôle de Scrum-master (animateur d'équipe) en assurant le bon déroulement du projet et en faisant la liaison entre le PO et notre équipe.
L'équipe en charge des deux projets était constituée de quatre personnes :

- Moi-même, jouant le rôle de Scrum-master et de responsable technique.
- Un graphiste.
- Un journaliste en charge de la rédaction des contenus.
- Une traductrice.

Nous avons employé les principaux outils de la méthode Scrum:

- Le product-backlog permettant de lister par ordre de priorité les exigences du product-owner.
- Le sprint-backlog présentant les tâches à réaliser durant les sprints.
- Une roadmap à l'aide d'un diagramme de GANTT pour visualiser les moments importants du projet.
- Le burndown chart pour connaitre l'état d'avancement des tâches dans le temps.
- Et enfin un task board pour voir rapidement les tâches à faire, en cours, à valider et réalisées.

Concernant les pratiques mises en œuvre, nous avons adapté Scrum aux spécificités de nos projets. Pour prendre l'exemple du projet CAPTOR, nous avons par exemple choisi de découper le projet en deux livraisons du site web, appelées ***releases***.

- Release 1 : Réalisation des travaux à risque (nombreux acteurs externes, dépassement des délais probable) : conception graphique, préparation du socle technique, réalisation d'interviews vidéo. Livraison du site web en version beta.
- Release 2 : Réalisation des travaux à haut niveau de maitrise (aucun acteur externe, délais maitrisés) : postproduction et intégration des contenus rédactionnels. Livraison du site web définitif.

Nous avons ensuite découpé chacune des deux releases en courtes itérations (les sprints) de trois semaines, qui correspondaient aux échéances clefs du projet et aux validations que nous voulions obtenir du PO. Nous avons aussi réalisé un sprint de sept semaines en raison de la période de

congés de notre PO. A la fin de chaque sprint, nous avons tenu des revues de sprint avec ce dernier.

2.2. Recueil des besoins

Tout projet a pour objectif de répondre aux exigences du client. Le recueil des besoins est la première phase essentielle d'une gestion de projet classique ou agile.
Selon Messager (2010), 19% des fonctionnalités d'un logiciel sont rarement utilisées et 45% ne le sont jamais ! Par ailleurs il constate que 56% des défauts logiciels ont pour origine une mauvaise compréhension des besoins.
Les projets avec lesquels nous avons expérimenté Scrum comportaient un cahier des charges très précis et détaillé ; c'est souvent le cas lorsque nous répondons à des appels d'offres, car ce document est un élément contractuel du marché. Ainsi, nous partons d'une bonne base pour le recueil des besoins. Toutefois, cela ne suffit pas, car comme nous avons pu en faire l'expérience sur d'autres projets, le client et l'équipe de développement ne parlent pas toujours le même langage et la problématique que pose Messager (2010) est la suivante : « *comment, moi, client, puis-je être assuré que mes besoins sont compris et comment nous, équipe de développement, pouvons être certains d'avoir compris le client ?* ».
Nous pouvons ajouter que même si nous travaillons dans l'agilité et que les besoins du client peuvent évoluer durant le projet, il est nécessaire de bien comprendre la vision du produit du client dès le départ. Des approximations dans la compréhension des besoins ternissent l'image de l'équipe et n'engagent pas le projet dans un rapport de confiance réciproque.

2.2.1. Comment recueillir efficacement les besoins ?

A cette question, les pratiques agiles apportent déjà un élément de réponse avec la mise en place d'un développement itératif. Il s'agit de faire des livraisons régulières du produit afin d'obtenir des retours rapides du client. Nous avons pu nous en rendre compte sur les deux projets, le fait de voir notre PO fréquemment et régulièrement nous a permis de nous ajuster durant le projet.
Mais cela n'est pas suffisant, il faut mettre en œuvre des techniques de recueil efficaces dès le lancement du projet. Pour cela, nous avons étudié la littérature agile, mais n'avons pas été convaincu par les solutions proposées qui nous ont paru soit trop vagues comme le brainstorming ou les forums de discussion, soit trop longues à mettre en œuvre (technique des neuf cases, observation du comportement des utilisateurs, questionnaires). Nous avons donc recherché d'autres approches et avons été séduit par les « *innovation games* » (Hohmann, 2006).

2.2.2. L'utilisation des « innovation games »

Les « *innovation games* » sont des jeux qui ne doivent pas être confondus avec les « *serious games* ». Alors que ces derniers ont une portée pédagogique, informative ou d'entrainement, les « *innovation games* » ont pour unique objectif de mieux comprendre les besoins des clients. Selon Hohmann (2006) « *innovation games are fun ways to collaborate with your customers to better understand their needs* ».

Il existe douze jeux répartis en quatre catégories permettant chacune de répondre à un de ces quatre objectifs :

- Déterminer les dysfonctionnements du système existant.
- Identifier les besoins essentiels du client.
- Identifier les craintes du client autour du produit ou du service à réaliser.
- Prioriser les fonctionnalités à développer.

On choisit donc un jeu en fonction de l'objectif que l'on souhaite atteindre, mais aussi selon six critères qui les caractérisent, parmi lesquels l'effort de préparation pour leur mise en place ou encore leur durée.

Dans le cadre de la mise en pratique de l'agilité avec nos clients, nous avons utilisé essentiellement deux jeux très efficaces, le « *speed boat* » et « *remember the future* ».

2.2.2.1. Le « speed boat »

Objectif : identifier les dysfonctionnements du système existant. Dans le cadre du projet mené avec le CRCT-INSERM cela nous a permis d'identifier ce que notre PO n'appréciait pas dans le site web actuel.

Déroulement : nous dessinons un bateau sur un tableau en expliquant que l'on souhaite qu'il aille vite. Puis nous demandons au PO de dessiner une ancre qui le freine, en positionnant dessus des post-it sur lesquels sont indiqués un dysfonctionnement. Plus le post-it est placé au bas de l'ancre, plus il symbolise un problème bloquant. L'ordre des post-it alimente la discussion avec le PO.

2.2.2.2. « Remember the future »

Objectif : identifier les besoins essentiels du client en faisant émerger sa définition du succès du projet.

Déroulement : ce jeu ne nécessite aucune préparation ; on demande au client de s'imaginer qu'il utilise notre produit quotidiennement depuis six mois (la durée peut être ajustée en fonction du temps d'utilisation qui semble le plus pertinent) et on lui demande de nous raconter en détail comment ce dernier a contribué à lui apporter satisfaction sur le dernier mois écoulé. Selon Hohmann (2006), des études de psychologie cognitive démontrent qu'il est plus facile à une personne de se « *projeter dans le passé* », plutôt que de lui demander d'imaginer ce dont il aura besoin dans le futur. Pour avoir essayé les deux tournures de phrases avec quatre clients différents, les résultats sont étonnants ! En utilisant la méthode de Hohmann (2006), nos PO sont plus précis dans les besoins qui leurs semblent essentiels et sont plus prompts à les énoncer. Nous avons eu des réponses inattendues lors du projet CRCT-INSERM, les besoins se sont centrés sur la nécessité de fournir des contenus rédactionnels de qualité plus que sur les fonctionnalités requises. Les contenus étant produits par le PO, ce dernier s'est ainsi étonné et nous a dit « *en fait c'est à nous de vous fournir les éléments les plus importants pour le succès du projet !* ». Cette pratique permet ainsi des prises de consciences utiles à l'avancée du projet.

2.2.2.3. Atouts des « innovation games »

L'utilisation de ces jeux nous a semblé efficace dans nos projets pour trois raisons :

- Chaque jeu est conçu pour atteindre un objectif précis et permettant de répondre aux problématiques de tout projet agile (et non agile), le recueil des besoins, la priorisation des exigences du PO, l'identification des risques…
- Ils sont rapides et simples à mettre en œuvre, c'est une condition essentielle lorsque notre client ne peut consacrer qu'une heure et demie à une réunion de lancement.
- Ce sont des jeux, nos clients prennent du plaisir à y participer et à s'y impliquer, cela permet aussi de créer une proximité bénéfique pour la suite du projet entre le client et l'équipe Scrum.

Nous avons utilisé les « *innovation games* » à divers moments des projets, au lancement ou lors de revues de sprint. Cela n'a jamais posé la moindre difficulté d'amener un PO à participer à un jeu, bien au contraire, lorsque nous disons lors d'une réunion de lancement que nous allons « *jouer* » pendant une demi-heure, cela contribue toujours à créer un climat de détente et à engager une relation moins stricte qu'à l'accoutumée.

2.3. Retour d'expérience

Nous faisons ici un retour d'expérience sur les projets pilotes concernant les pratiques utilisées puis les outils employés.

2.3.1. Concernant les pratiques

2.3.1.1. La présentation de la démarche agile au product-owner

Dans les projets pour lesquels nous avons mis en œuvre l'agilité, nous n'avons jamais voulu imposer notre démarche. Il a donc été essentiel de présenter à nos clients la façon dont nous souhaitions mener leurs projets dans l'agilité. Nous profitons toujours de la réunion de lancement pour expliquer notre approche. Nous présentons les grandes valeurs de l'agilité sans pour autant, bien sûr, réciter le manifeste. Il s'agit surtout de faire comprendre au client deux points qui nous paraissent essentiels :

- Nous souhaitons réaliser un produit qui réponde à leurs besoins le plus précisément possible.
- Il est essentiel qu'il y ait une implication significative de leur part durant le cycle de vie du projet.

Nous expliquons ensuite les rôles, les pratiques et les outils que nous allons employer. Concernant les outils, nous avons déjà préparé tous ceux que nous utiliserons durant le projet, en prenant le risque que le client refuse de travailler dans l'agilité. Cela nous permet en effet de leur présenter un élément concret avec lequel nous pouvons déjà commencer à discuter. En effet, la présentation d'un premier product-backlog nous permet par exemple de vérifier l'ensemble des exigences du client.

Qu'avons-nous pu constater concernant cette étape importante de présentation de la démarche ?

Nous avons été surpris de voir à quel point nos interlocuteurs nous font confiance lorsque nous présentons la démarche agile. Il y a en effet très peu de négociation autour des exigences que nous recensons et des priorités de développement, ainsi que des principales échéances proposées. Il faut toutefois nuancer ces propos car nous travaillons essentiellement dans le cadre de marchés publics avec des cahiers des charges précis.

Sur le projet CAPTOR, notre interlocuteur nous a d'ailleurs dit qu'il appréciait notre gestion de projet qui lui paraissait très structurée et rassurante, la présentation de la démarche établit donc un climat de confiance dès le départ.

Par ailleurs, j'ai été agréablement surpris de constater que le PO emploie très facilement la terminologie agile qu'il découvre à peine en utilisant immédiatement les termes anglophones de « *release* » et de « *sprint* » dans la conversation.
Il est à noter pour finir que la préparation et le suivi des outils prend du temps, il est primordial de mettre à jour ces derniers au fur et à mesure pour ne pas perdre le fil du projet. C'est d'ailleurs une des raisons pour lesquelles nous avons décidé de concevoir l'application que nous allons présenter dans la suite du mémoire en page 75.

2.3.1.2. La réalisation d'un sprint

Rappel : dans une gestion de projet agile, le projet est découpé en plusieurs étapes d'une durée identique (en général quelques semaines), nommées « *itérations* » ou « *sprints* » dans le cadre de Scrum. A la fin de chaque itération, lors de la revue de sprint, une version fonctionnelle du produit est présentée au product-owner (Messager & Tabaka, 2010).

La contractualisation dont nos projets font l'objet lors des marchés publics nous pousse à devoir adapter certaines pratiques agiles, nous sommes en effet contraints de répondre aux délais et aux exigences du cahier des charges initial sous peine de pénalités financières. Ainsi, dans le cadre théorique d'une méthode agile Scrum, le PO est censé pouvoir arrêter le projet à l'issue d'une revue de sprint, s'il juge le produit répondant à ses besoins hors dans notre contexte, nous devons réaliser toutes les exigences initiales. Toutefois pour chaque sprint, nous avons une certaine latitude pour choisir celles qui seront développées, nous les avons définies au lancement du projet par ordre de priorité et nos PO nous font confiance sur ce point.

2.3.1.2.1. La durée des sprints

Scrum préconise des sprints d'une durée comprise entre deux et quatre semaines. Sur les deux projets, nous avons dû mener à un moment donné un sprint plus long en raison des contraintes de nos PO (congés, impossibilité d'être présent…). Cela a tendance à casser le rythme du projet et se ressent lors des revues de sprint qui suivent, le PO aura plus de difficultés à suivre les éléments du projet qu'il doit nous transmettre et son implication doit éventuellement être stimulée à nouveau. Toutefois il est difficile en pratique de mettre en œuvre des sprints d'une durée parfaitement égale, l'idée est de ne pas mener des itérations avec des différences de durées trop importantes (plusieurs semaines).

2.3.1.2.2. La communication et les échanges lors d'un sprint

Il est important de préciser que le rôle du scrum-master (que je joue) est crucial dans la motivation aussi bien de l'équipe que du PO. Nous avons pu voir qu'il était important de réexpliquer parfois la démarche agile et son rôle au PO, afin que ce dernier soit toujours impliqué et qu'il ne retombe pas dans les travers des méthodes classiques, notamment rejeter sur l'équipe de développement toute la responsabilité des échéances à tenir.

Par ailleurs, au cours d'un sprint durant lequel nous devions intégrer des contenus rédactionnels dans le site web, le PO nous a mis en rapport direct avec les personnes chargées de la rédaction de ces derniers. Cela a été compliqué à gérer pour nous car d'une part, nous ne les connaissions pas et d'autre part, nous avons dû les relancer à de nombreuses reprises. Nous avons fait en sorte d'éviter cette situation lors des autres projets car il est nécessaire que le PO soit notre seul interlocuteur pour trois raisons :

- Cela le responsabilise et lui permet d'être parfaitement au courant de l'avancée du projet.
- Le scrum-master ne possède pas les mêmes « *leviers* » pour dialoguer avec des interlocuteurs que connait en général le PO.
- Lors de la prise de décision sur un élément du projet, il est important d'avoir une seule personne pour transmettre à l'équipe Scrum la décision qui fait consensus.

Le schéma suivant permet de résumer cette idée selon laquelle le PO doit être l'interlocuteur unique du scrum-master.

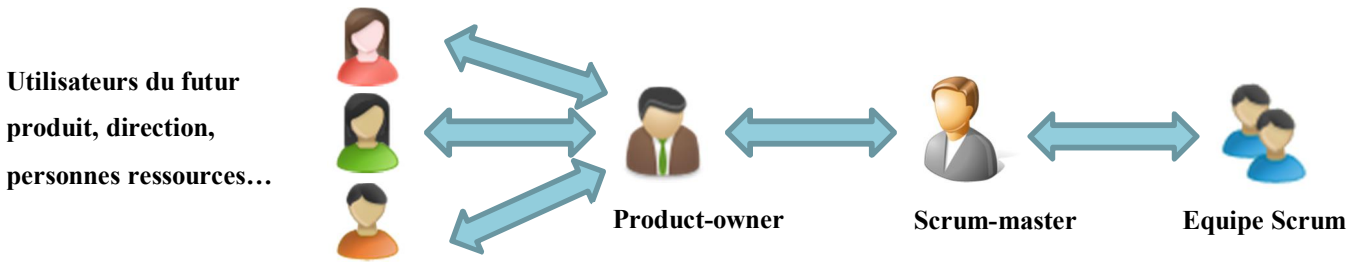


Figure 13 : Représentation pour des échanges optimaux lors d'un projet Scrum.

2.3.1.3. La revue de sprint

Rappel : la fin d'un sprint donne lieu à une version partielle mais opérationnelle du produit qui est présentée au PO lors d'une réunion limitée à quatre heures pour une itération d'un mois (Fernandez et al., 2013).

Nous invitons systématiquement notre PO dans nos locaux pour deux raisons ;

- Cela permet d'impliquer notre PO en l'intégrant au sens propre à notre équipe et en l'immergeant dans notre environnement.
- Le PO se retrouve dans un cadre extérieur à sa structure et nous essayons de créer les conditions permettant de passer un moment de dialogue agréable et constructif (un café est souvent un bon début !).

Nous nous sommes inspiré des étapes préconisées par Aubry (2013) pour préparer et mener nos revues de sprint. Elles s'articulent de la façon suivante:

1. Préparation de la revue de sprint : impression du product-backlog, du sprint-backlog, du burndown chart et de la roadmap dans sa dernière version.
2. Rappel du principe de la méthode Scrum au PO et point global sur l'avancée du projet avec les outils cités précédemment.
3. Rappel des objectifs du sprint.
4. Démonstration : nous présentons les user stories (exigences) terminées.
5. Evaluation des résultats du sprint : le PO et un intervenant nous posent des questions s'ils le souhaitent et nous donnent leurs premières impressions sur notre travail.
6. Nous faisons un point avec le PO sur les exigences qui sont prioritaires et que nous allons développer lors du sprint suivant et définissons avec lui la date de notre prochaine revue de sprint.

Après le départ du PO, nous faisons une **rétrospective du sprint** avec mon associé pour débriefer le rendez-vous et nous envoyons en suivant un compte rendu de notre entrevue au PO. Nous affinons ensuite les tâches qui seront à réaliser afin de répondre aux exigences du PO.
Lors d'un projet et contrairement à ce que nous préconise Scrum, nous avons fait l'erreur d'anticiper sur les tâches que nous pensions réaliser dans les sprints à venir et de lui en parler. Effet immédiat, le PO avait acté mentalement que nous réaliserions certaines exigences plus tôt que prévu, nous contraignant ainsi à de nouvelles échéances sans que nous soyons sûrs de les tenir. En bref, nous sommes retombés en partie dans le schéma d'une méthode de gestion de projet classique en oubliant les principes fondamentaux de l'agilité…

2.3.1.4. La mêlée quotidienne

Rappel : selon Fernandez et al. (2013) la mêlée quotidienne

> (ou daily Scrum meeting) permet à l'équipe Scrum de faire un « *point de coordination sur les tâches en cours et sur les difficultés rencontrées. Elle dure quinze minutes […], le PO n'assiste pas à cette réunion […], le scrum-master pose trois questions à chaque membre de l'équipe : qu'as-tu fait hier ? Quelles difficultés as-tu rencontrées ? Et que vas-tu faire aujourd'hui ?* »

Nous sommes entre trois et quatre personnes à participer à la mêlée quotidienne : mon associé, moi-même, un stagiaire selon la période et un journaliste web appelé en soutien en fonction du projet. La mise en place de cette pratique au sein de CosiWeb est un changement majeur dans notre gestion de projet, elle a plusieurs effets bénéfiques :

- Cela nous permet d'être parfaitement au courant de ce sur quoi chaque membre de l'équipe travaille.
- Cela nous permet aussi de détecter et résoudre rapidement les problèmes lorsqu'ils apparaissent.

Il faut de la **volonté** pour s'imposer de mener chaque jour ces réunions car nos tâches quotidiennes peuvent nous pousser à nous dire que nous avons des choses plus urgentes à faire. Il faut aussi de la **franchise** et du **courage** pour parler des problèmes que nous rencontrons.
J'ai aussi noté un effet pervers dont je peux parfois être à l'origine en tant que Scrum-master, la mêlée quotidienne peut parfois avoir tendance à se transformer en une liste de reproches effectués à l'équipe. Hors, il faut s'efforcer à faire de ce moment un échange constructif.

2.3.2. Concernant les outils

2.3.2.1. Les outils collaboratifs

Depuis son lancement en 2012, nous utilisons au sein de CosiWeb le logiciel Google Drive, un service de stockage « *dans le cloud* » qui nous permet de stocker et de partager des fichiers de tous types avec nos clients. Il est par ailleurs couplé avec Google Docs qui est une suite bureautique web nous permettant entre autres de pouvoir élaborer des questionnaires auprès de nos clients (l'un d'entre eux sera d'ailleurs étudié dans la suite du mémoire en page 95).
De façon systématique, nous mettons en place dans Google Drive un espace pour chaque projet que nous menons dans l'agilité. Cela représente trois intérêts :

- Notre PO et l'équipe en charge du projet peuvent échanger des ressources dans un espace collaboratif commun et consultable depuis n'importe où.

- Le PO peut consulter à tout moment l'avancée du projet grâce aux outils Scrum qui sont disponibles.
- Google Drive offre une sauvegarde automatique dans le cloud des documents que nous déposons dessus.

L'espace collaboratif possède l'arborescence suivante :

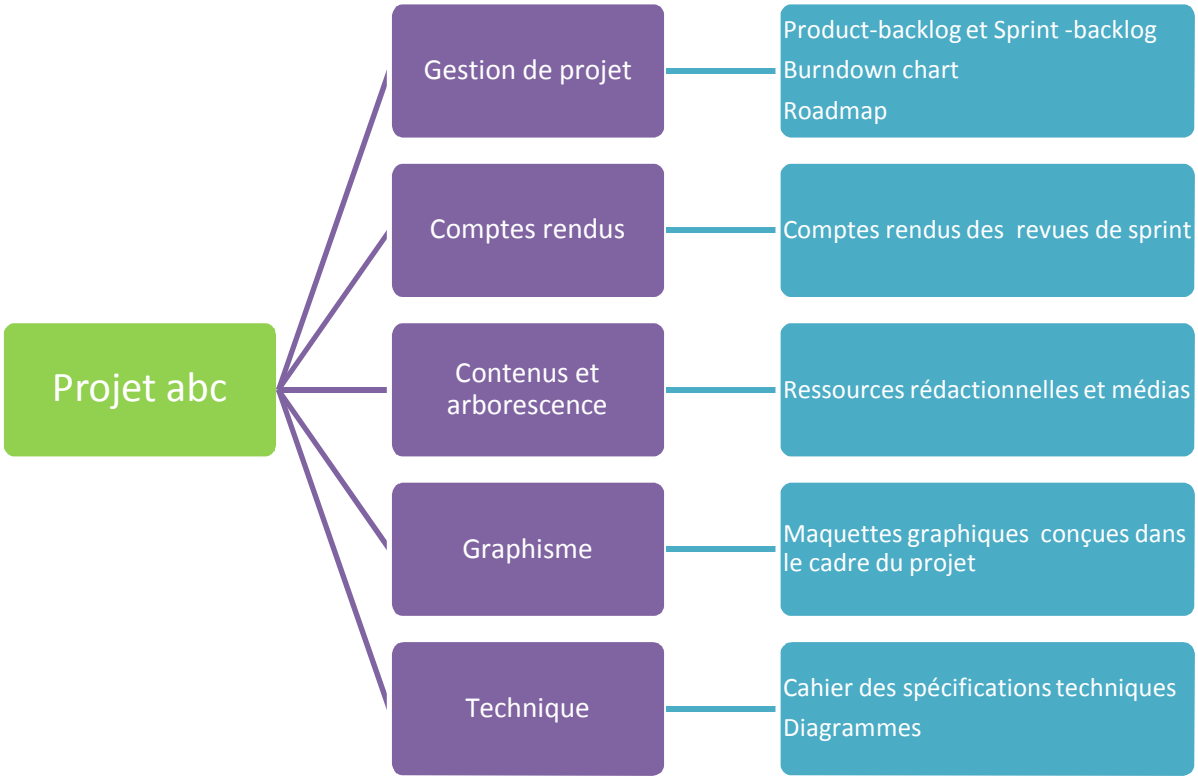


Figure 14 : Arborescence de l'espace collaboratif mis en place pour nos clients dans le cadre d'un projet agile.

Il est à noter que nous centralisons absolument tous les documents dans cet espace.
Hélas, durant les projets que nous avons menés, nous nous sommes rendu compte que peu de nos PO utilisaient l'espace collaboratif pour deux raisons :

- Certains sont frileux à l'idée d'installer le logiciel, parfois car le service informatique de leur structure y est réticent.
- L'utilisation des emails est encore très ancrée dans les habitudes des utilisateurs pour échanger des documents malgré toutes les limites que cela impose : limitation du poids des fichiers, problème de gestion des versions…

Lorsqu'un PO ne veut pas utiliser cet espace, il perd surtout le bénéfice de consulter l'avancée du projet en quasi temps réel. Dans tous les cas nous utilisons la plateforme plusieurs fois par jour

au sein de l'équipe pour échanger des ressources, faire un point sur le suivi du projet et bénéficier de la sécurité de sauvegarde des données dans le cloud.
Par ailleurs, lorsque le PO n'utilise pas l'espace, nous nous occupons d'y regrouper les ressources qu'il nous transmet et après chaque revue de sprint nous lui transmettons par email les outils lui permettant de suivre l'avancée de son projet : le product-backlog, le sprint-backlog et le burndown chart.
Ainsi malgré le succès mitigé de cet espace collaboratif auprès de nos PO, il conserve toutefois un intérêt indéniable pour rationaliser la gestion des ressources d'un projet et les centraliser dans un espace sécurisé.

2.3.2.2. Le product-backlog

Rappel : le **product-backlog** (carnet de produit) permet de lister les exigences données par le product-owner et classées par priorités. Ces exigences sont appelées des user stories (récits d'utilisateur ou scénarios) et décrivent une fonctionnalité à développer.

Lors de la mise en place des outils, nous avons suivi le principe « *KISS* » (Keep it Simple, Stupid), autrement dit « *ne compliquons pas les choses* ». Nous aurions pu en effet décider d'installer une plateforme web comme iceScrum[15] pour que nos clients puissent avoir un suivi de nos projets, mais nous n'avons pas opté pour cette solution car nous voulions éviter d'ajouter une formation à nos PO en début de projet pour utiliser une solution comme iceScrum.
Nous avons ainsi décidé de créer ces outils à l'aide d'Excel pour plusieurs raisons :

- C'est un logiciel que tous nos PO possèdent.
- Excel est souple et permet de créer des outils adaptés à nos projets.
- Nous pouvions facilement et rapidement mettre en place ces outils.
- Ces derniers peuvent être déposés dans l'espace collaboratif que nous avons mis en place et facilement accessibles par nos PO.

Par ailleurs, nous avons choisi de profiter de l'expérience acquise sur ces projets pilotes afin de développer en parallèle l'application ASIPAT (Application de Suivi Individuel de Projets Agiles

[15] http://www.icescrum.org/

et de Tests fonctionnels) que nous présenterons en page 75. Il n'était donc pas pertinent de nous lancer dans la mise en place d'une autre solution de suivi.

Le product-backlog se présente de la manière suivante :

PRODUCT BACKLOG - PROJECT CAPTOR				
Release	Sprint	Titre	Item (User story)	Domaine
1	1	Architecture et arborescence du site	En tant que commanditaire, je veux une proposition d'architecture (zoning) et d'arborescence du site, afin de pouvoir définir l'ergonomie du site.	Ergonomie
1	1	Réalisation de la proposition graphique	En tant que commanditaire, je veux une proposition graphique sous forme de maquette, afin de pouvoir définir l'aspect visuel du futur site.	Graphisme

Figure 15 : Aperçu d'un product-backlog mis en place dans le cadre de projets agiles à CosiWeb.

Comme expliqué précédemment, nous faisons le point avec le PO à chaque revue de sprint en lui présentant le product-backlog, que nous pouvons réajuster à ce moment-là. En pratique, nous nous sommes rendu compte que nous le modifions peu jusqu'à la fin du projet. Une fois encore, cela provient du fait que nous répondons à des marchés pour lesquels les exigences (traduite par les users-stories du product-backlog) sont bien définies dès le départ.

Il manque dans nos product-backlogs une unité d'œuvre appelée « *points de récits* » et permettant de déterminer la complexité d'une exigence à développer. Cette unité d'œuvre prend en général les valeurs de la suite de Fibonacci afin de distinguer plus facilement l'ordre de complexité des exigences : 1, 2, 3, 5, 8, 13…

Déterminer la complexité d'une exigence permet d'associer à son développement le ou les membres de l'équipe possédant le niveau technique le plus adapté. Cela permet aussi de définir avec le PO le nombre de points de récit que l'on développera dans un sprint.

Pourquoi n'avons-nous pas mis en place cet indicateur ? Principalement car je suis le seul développeur de l'entreprise et que je n'ai pas jugé utile de le voir figurer. C'est une erreur car j'ai pu me rendre compte en discutant avec mes PO que c'est une information qui les intéresse, d'autre part, cela permet de faire comprendre à ce dernier le niveau de complexité de ces exigences. Ainsi, nous avons prévu d'ajouter cette unité d'œuvre dans l'application ASIPAT que nous avons développée.

2.3.2.3. Le sprint-backlog et le burndown chart

Rappel : les user stories du product-backlog sont traduites et répertoriées dans le **sprint-backlog** (carnet de l'itération) sous forme de tâches à réaliser au cours d'une itération.

Même si ce n'est pas fréquent, des tâches du sprint-backlog peuvent être revues et nous avons pu être amenés à en ajouter durant un projet, en fonction de nos échanges avec le PO.

SPRINT BACKLOG - PROJECT CAPTOR

Tâche	Estimation initiale	Semaine 1	Semaine 2	Semaine 3	Semaine 4	Semaine 5	Semaine 6
Réalisation de la proposition ergonomique : zoning et arborescence	2	0	0	0	0	0	0
Réalisation de la proposition graphique	5	5	5	1			
Conception des questions des interviews	9	9	9	0	0	0	0
Repérage des interviews vidéo	1	1	1	0			
Dépôt des domaines et préparation de l'hébergement	1	1	1	1			
Préparation et réalisation des interviews vidéo	15	15	15	15			
Configuration du moteur du CMS	5	5	5	5			
Intégration du graphisme sur le CMS	5	5	5	5			
Intégration de l'arborescence sur le CMS	2	2	2	2			
Préparation de la gestion bilingue	2	2	2	2			
Restant à accomplir	47	45	45	31			
Théorique	47	39	31	24	16	8	0

Figure 16 : Aperçu d'un sprint-backlog mis en place dans le cadre de projets agiles à CosiWeb.

Pour chaque tâche, nous définissons un nombre de jours-homme nécessaire pour la réaliser. Chaque fin de semaine, nous inscrivons le nombre de jours restants pour réaliser la tâche. La somme des jours restant pour réaliser l'ensemble des tâches nous permet de connaitre le volume de travail restant à accomplir jusqu'à la fin du sprint. Nous utilisons ces données afin de réaliser un graphique de progression, le burndown chart, qui permet de visualiser l'évolution de la quantité de travail restant. C'est un élément que nous présentons au PO lors des revues de sprint.

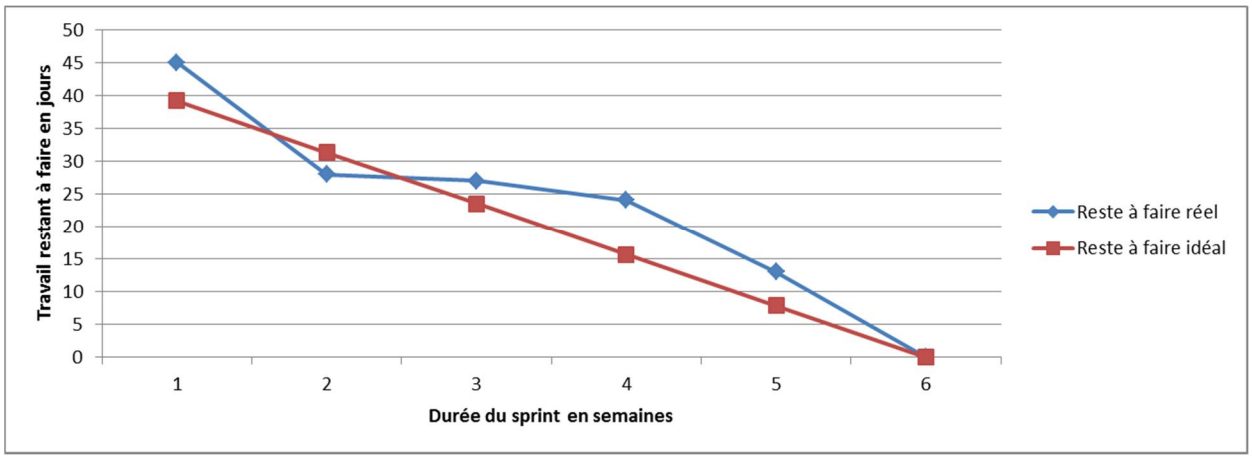


Figure 17 : Aperçu d'un burndown chart mis en place dans le cadre de projets agiles à CosiWeb.

Dans l'exemple ci-dessus, nous visualisons l'écart entre la courbe idéale de réalisation des tâches lors d'un sprint de six semaines et la courbe du travail réellement accompli par l'équipe Scrum.

On peut constater que le projet s'est déroulé plus vite que prévu en semaine deux mais que l'équipe a été moins performante par la suite. Elle a toutefois réussi à rattraper son retard en fin de Sprint.

2.3.2.1. La roadmap et le task board

> **Rappel :** la ***roadmap*** est un planning qui permet d'identifier les moments clefs de l'ensemble du projet. Le **task board** se présente sous la forme d'un tableau sur lequel on liste dans une première colonne les user stories à l'aide de post-it, et les fonctionnalités du sprint-backlog dans quatre colonnes pour représenter leur avancement : à faire, en cours, à valider, fait.

Nous l'avons vu précédemment, dans le cadre d'une démarche agile, il est proscrit de planifier toutes les tâches à réaliser pour éviter de tomber dans la rigidité d'un cycle en cascade. Il est toutefois utile d'effectuer une planification sur l'ensemble du projet pour déterminer les principales phases du projet. Cela nous a permis d'anticiper le moment où nos sous-traitants devaient nous livrer leurs travaux (de rédaction et de traduction par exemple) et quand le PO devait être disponible pour valider des points essentiels des projets. Bien entendu, nous sommes dans une approche agile et la roadmap est mise à jour après chaque revue de sprint. Nous réalisons la roadmap à l'aide d'un diagramme de GANTT.

Pour finir, nous utilisons pour chacun des projets que nous gérons un task board (tableau des tâches) aussi appelé Scrum board. C'est un outil basique, mais qui nous est essentiel pour visualiser rapidement l'avancement du projet. Lors des revues de sprint, nos PO vont souvent le consulter et nous ont dit être rassurés de voir que nous suivons de près leur projet. C'est aussi une base sur laquelle on peut échanger pour redéfinir ou préciser des tâches à réaliser.

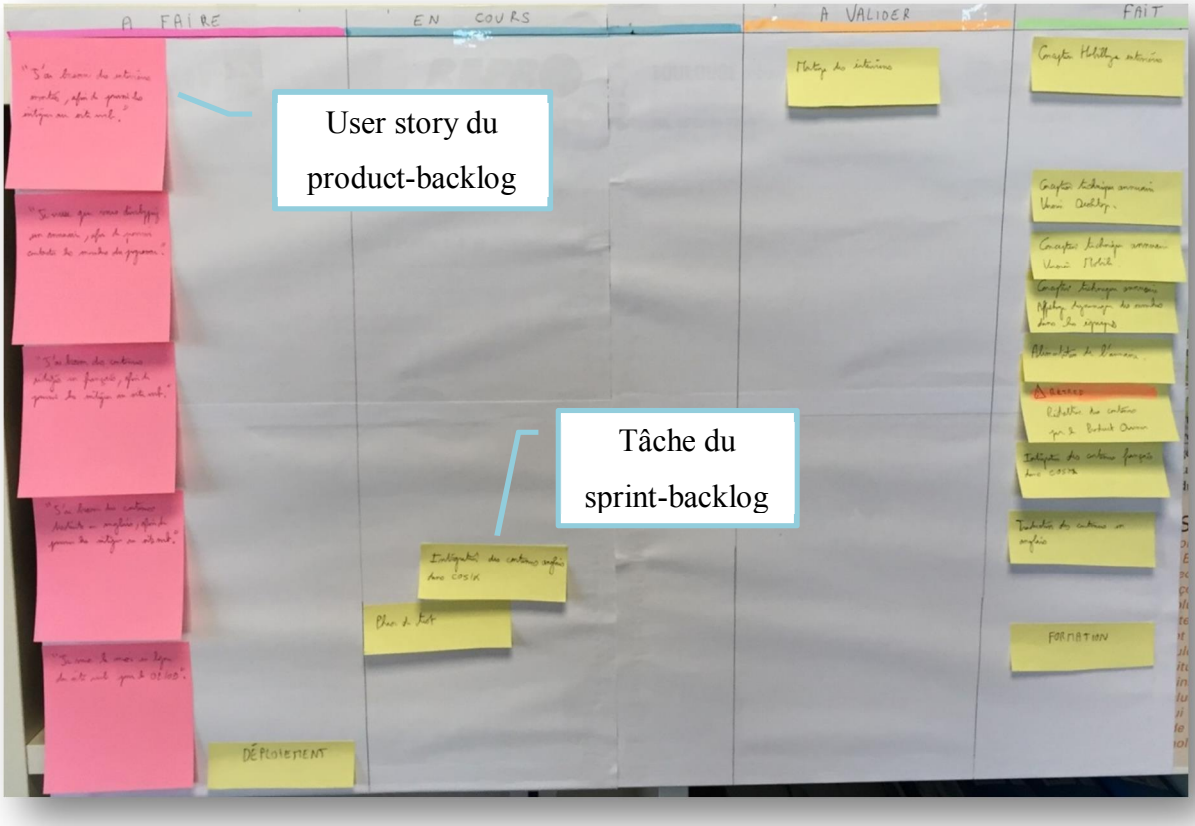


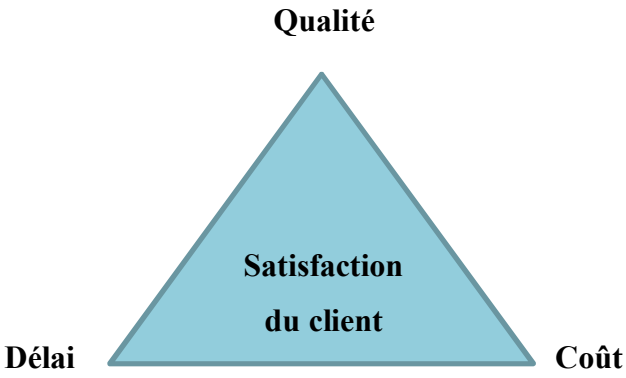
Figure 18 : Aperçu d'un task board mis en place dans le cadre de projets agiles à CosiWeb.

Nous avons présenté notre retour d'expérience des pratiques et des outils de la méthode Scrum. Quel bilan pouvons-nous en tirer ?

3. Bilan de l'utilisation des pratiques de Scrum

3.1. Quels ont été les bénéfices en matière de qualité, coût et délais ?

Les deux projets pilotes menés selon une démarche agile ont-ils été un succès ?
Si nous répondons à cette question en la faisant passer par le filtre du triangle de la performance vu précédemment, nous pouvons répondre, « *en majeure partie* ».
Il y a eu un **dépassement des délais** sur le projet CAPTOR en raison du retard du PO à valider certains points à la toute fin du projet, le projet CRCT-INSERM a quant à lui été réalisé dans les temps impartis.
Nous avons totalement répondu aux besoins de nos clients en atteignant l'ensemble de leurs exigences, c'est le **respect de la qualité**.
Enfin, **les coûts** ont été **maitrisés** et les projets ont été rentables pour CosiWeb.



Triangle de la performance

Un cycle en cascade aurait-il apporté de meilleurs résultats ?
Nous pensons que non. En effet la démarche agile a apporté au moins trois bénéfices importants pour notre équipe et que nous pourrons reproduire dans nos projets agiles futurs :

- **Une excellente vision du projet à tout moment** : nous pouvons à tout moment nous repérer dans l'avancée du projet grâce aux outils Scrum. Par ailleurs, nous travaillons sur plusieurs projets en parallèle et il est aisé de reprendre en main un projet Scrum sur lequel nous n'avons pas travaillé pendant deux semaines.
- **Une meilleure communication au sein de notre équipe** : grâce à la mêlée quotidienne, nous avons une vision plus claire de nos travaux respectifs et des contraintes que cela implique pour chacun de nous.
- **Une très bonne implication des PO** : ils sont totalement rentrés dans le jeu lors des deux projets.

Un problème que nous rencontrons parfois lors des projets non agile est une perte d'implication du client (même si c'est le financeur et le destinataire du projet), couplée à une décharge des responsabilités sur notre équipe en cas de retard ou de non-conformité aux exigences. L'approche agile propose les pratiques et les outils permettant une collaboration importante avec le PO.

Toutefois, l'adhésion à ces pratiques n'est pas évidente, à la fois pour l'entreprise et le PO, et elle s'obtient en étant honnête et parfois courageux quand il faut dire que quelque chose ne fonctionne pas. Nous nous sommes aussi rendu compte que ces pratiques nous permettaient de faire preuve de plus de transparence envers le PO.

3.2. Les besoins initialement constatés sont-ils comblés ?

Nous allons reprendre les dysfonctionnements constatés au sein de CosiWeb et relevés en page 35, et voir comment les mesures palliatives mises en place avec Scrum les ont comblés.

> **Rappel** : nous nous sommes appuyé sur la méthode Méhari d'analyse des risques. Les dysfonctionnements que nous listons ci-dessous sont ceux qui peuvent mener à l'échec partiel ou total d'un projet, dans le cadre d'une gestion de projet classique menée au sein de la société CosiWeb.

Etape du projet	Dysfonctionnement constaté avant la mise en pratique de Scrum	Pratique Scrum palliative mise en place
Formulation des besoins du client	Pas de prise en compte de tous les besoins du client.	Elaboration d'un ***product-backlog*** listant l'ensemble des exigences du PO.
Recueil des besoins du client	Mauvaise compréhension des besoins du client.	***Revues de sprint*** régulières avec le PO.
Planification	Pas de priorisation des développements.	Choix des fonctionnalités à développer durant les ***rétrospectives de sprint*** et reports dans le ***sprint-backlog***.
Planification	Dépassements des délais.	Mise en place d'un ***burndown chart*** mis à jour fréquemment afin de vérifier l'avancement des tâches.

Communication au sein de l'entreprise	Les retours clients ne sont pas toujours formalisés et relayés entre les gérants de l'entreprise. Perte d'information.	***Mêlées quotidiennes*** au sein de l'équipe.
Communication avec le client	Effet tunnel sur certains projets.	***Revues de sprint*** régulières avec le PO. Validation régulière des exigences du ***product-backlog***.
Suivi du projet	Le planning prévisionnel n'est plus mis à jour une fois le projet en phase de pilotage.	Tenue d'une ***roadmap*** mise à jour régulièrement. Mise en place d'un **task board**.
Suivi du projet	Absence de reporting de pilotage permettant de mesurer l'écart entre le réalisé et l'attendu.	Mise en place d'un ***burndown chart*** mis à jour fréquemment.
Validation des travaux	Pas de validation des fonctionnalités développées. Demandes incessantes du client après le déploiement du produit.	***Revues de sprint*** régulières avec le PO. Validation régulière des exigences du ***product-backlog***.

Tableau IX : Besoins de CosiWeb et pratiques Scrum les ayant comblés.

Il reste toutefois des besoins que nous avions détectés et qui n'ont pas été comblés, ce sont ceux relatifs aux pratiques d'ingénierie logicielle, avec l'absence de tests unitaires et fonctionnels.

Etape du projet	Dysfonctionnement constaté	Pratique agile palliative
Conception du produit	Pas de *tests unitaires*, constatation tardive de bogues, régression.	Pratique des **tests unitaires** dans le cycle de développement.
Conception du produit	*Tests fonctionnels* réalisés manuellement et incomplets. Constatation tardive de bogues.	Rationnaliser les **tests fonctionnels**.

Tableau X : Besoins de CosiWeb non comblés par Scrum.

Bien entendu, des tests sont quand même réalisés par l'équipe et à de nombreuses reprises, mais il s'agit de tests effectués « *à la main* », qui sont ainsi menés de façon empirique par l'équipe en charge du projet en constatant le fonctionnement apparent de l'application dans un navigateur Internet. En pratiquant de la sorte, il est évident que tous les cas de figures peuvent difficilement être pris en compte. Il arrive parfois que des bogues soient constatés par le client ou l'équipe après livraison du produit.

Pour répondre en partie à ces besoins, nous avons choisi de développer une application qui permet d'une part à nos clients de suivre l'avancée de leur projet mené dans l'agilité et qui d'autre part, fournit au développeur une plateforme locale de suivi et de gestion de tests fonctionnels.

4. Conception de l'application ASIPAT (Application de Suivi Individuel de Projets Agiles et de Tests fonctionnels)

Nous allons tout d'abord faire une présentation générale d'ASIPAT (Application de Suivi Individuel de Projets Agiles et de Tests fonctionnels) avant de rentrer dans le détail de son fonctionnement.

4.1. Présentation générale de l'application

ASIPAT répond à deux objectifs :

- Permettre à l'équipe de CosiWeb de jouer et de gérer ses tests fonctionnels.
- Informer le PO de l'avancée de son projet mené dans l'agilité, et le tenir au courant de l'état des tests fonctionnels réalisés pour vérifier ses exigences.

ASIPAT est un système constitué de deux applications web qui dialoguent entre elles, une locale et une distante:

- **L'application web locale** est utilisée par l'équipe de CosiWeb (uniquement) et en particulier le développeur, pour gérer et jouer des tests fonctionnels ou des scripts de navigation.
- **L'application web distante** est consultable par les PO afin de connaitre l'avancée du projet et le résultat des tests fonctionnels.
- Le dialogue entre les deux applications s'effectue à l'aide d'un **service web**.

Remarque : à l'heure actuelle, je suis le seul développeur capable de réaliser des tests fonctionnels au sein de CosiWeb. C'est pour cette raison qu'il n'y a qu'une instance de l'application locale. Toutefois, ASIPAT a été conçu de telle façon qu'il sera possible à plusieurs testeurs de travailler en parallèle, en installant une application locale sur chacun de leurs postes. Cela permettra à chacun de gérer ses propres scripts, puis de les rattacher le moment venu aux exigences enregistrées dans l'application distante. Dans un souci de simplicité, nous présenterons dans la suite du document l'architecture d'ASIPAT avec une seule application locale car c'est le fonctionnement actuel mis en œuvre.

ASIPAT a été développé à partir du CMS CosiX conçu par mes soins. Voici une brève présentation des principales technologies employées pour son développement:

- Le langage de programmation **PHP** dans sa version 5.5.
- Le système de gestion de bases de données **MySQL** en version 5.6.

- **CasperJS** couplé avec le navigateur « *headless* » **PhantomJS** pour lancer les scripts de tests fonctionnels et de navigation.
- Le **protocole SOAP** (Simple Object Access Protocol) pour assurer le service Web entre l'application locale et l'application distante.
- Les objets **XMLHttpRequest** et **ActiveXObject** pour les traitements asynchrones.

L'architecture **WAMP** (Window, Apache, PHP, MySQL) a été adoptée pour le fonctionnement de l'application locale et **LAMP** (Linux, Apache, PHP, MySQL) pour celui de l'application distante.

4.2. Fonctionnement de l'application locale et distante

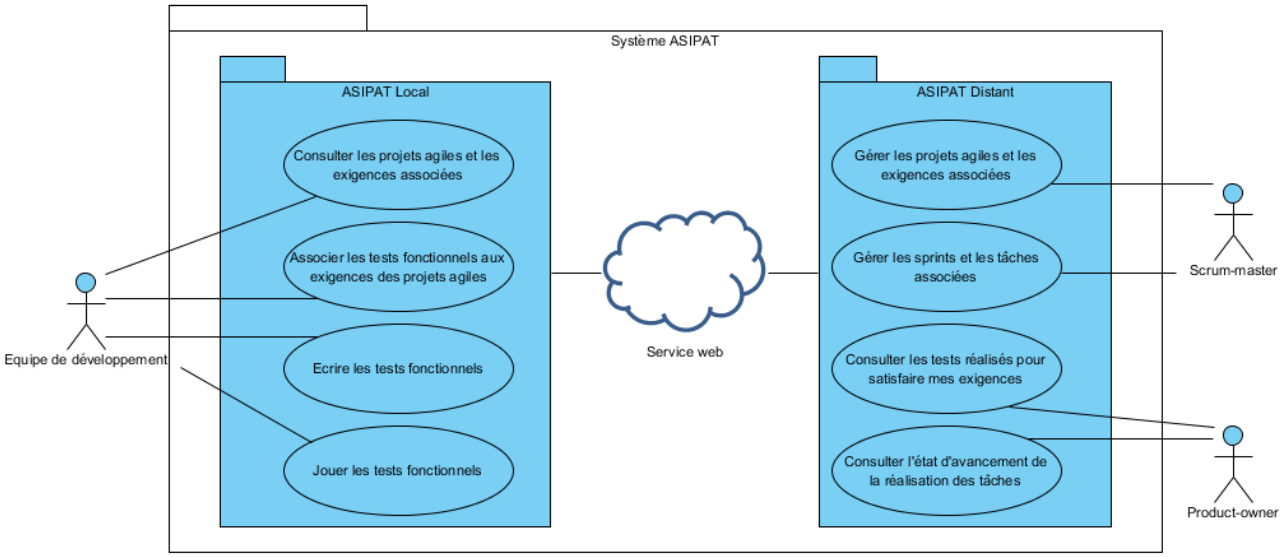


Figure 19 : Cas d'utilisation général du fonctionnement de l'application ASIPAT locale et distante.

Remarque : l'application locale est capable de gérer des scripts de test et de navigation et de les lancer grâce à l'utilitaire CasperJS[16] que nous détaillerons plus loin. Un script de test va permettre de tester une exigence grâce à des méthodes d'assertion, alors qu'un script de navigation va simuler le comportement d'un internaute dans un site web.

Dans l'application locale qui est utilisée par l'équipe de développement :

- On peut écrire des scripts de navigation ou de test fonctionnels et les jouer.
- Ces tests peuvent être enregistrés en base de données pour les rejouer par la suite.

[16] http://casperjs.org/

- Ces tests peuvent être associés aux exigences (qui sont en fait les users stories du product-backlog) gérées dans l'application distante.
- L'application locale consomme des services web pour afficher les projets, exigences et résultats de tests fonctionnels enregistrés dans l'application distante.

Dans l'application distante qui est utilisée par le Scrum-master et le PO :

- Le Scrum-master peut gérer les comptes des PO et leur donner un accès au suivi de leur projet.
- Le Scrum-master peut gérer les projets, les exigences, les tâches et les sprints.
- Le PO peut connaitre à tout moment l'avancée de son projet agile en consultant l'état d'avancement des tâches par l'équipe de développement.
- Le PO peut consulter le résultat des tests fonctionnels qui est envoyé par l'application locale.

4.3. Pourquoi développer une application autour des tests fonctionnels ?

Cohn (2009) décrit les différents niveaux de tests qui doivent être menés dans une démarche agile. Il les représente au moyen d'une pyramide des tests.

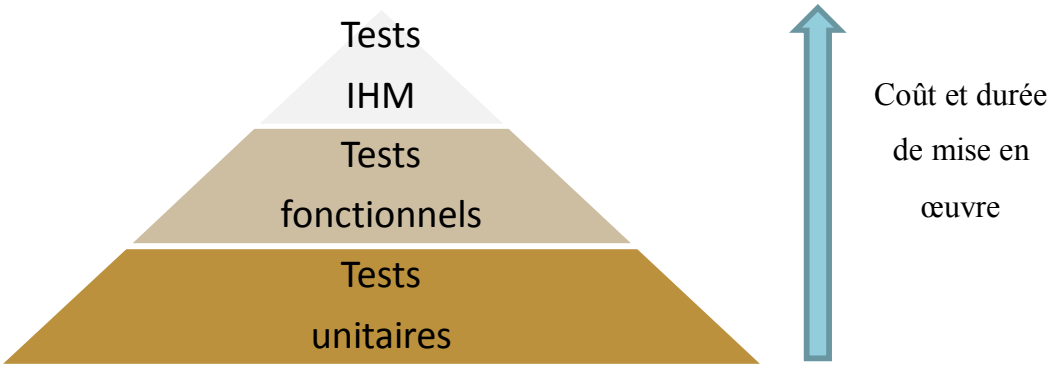


Figure 20 : La pyramide des tests pour projets agiles (Cohn, 2009).

Les tests unitaires sont la base de l'édifice, ils sont assurés par l'équipe de développement et consistent à vérifier le comportement de chaque morceau de code selon Vernois (2014).
Ils apportent plusieurs bénéfices :

- Ils permettent de déceler des problèmes de façon plus précoce et ainsi de diminuer les coûts de leur résolution.
- Les tests deviennent une documentation de ce que réalise un code source.
- Ils sont théoriquement moins coûteux à écrire que les tests fonctionnels.

Dans le cadre de notre activité de création de sites web, ce dernier point est selon moi plus contestable car ASIPAT permet d'écrire et de générer des tests fonctionnels de façon simple (à l'aide notamment de Resurrectio et de CasperJS que nous présenterons) et de surtout pouvoir les rejouer plusieurs fois dans chaque itération du projet afin de s'assurer que le système fonctionne toujours correctement.

Les tests fonctionnels vérifient l'adéquation entre les exigences du PO et les développements réalisés. Selon Legeard (2014), ils permettent aussi de « *garantir la non régression applicative associée aux évolutions (ce qui fonctionnait doit toujours fonctionner)* ». Des tests fonctionnels bien menés contribuent à éviter l'effet tunnel : un produit livré non conforme avec les besoins du client.

Enfin, nous retrouvons tout en haut de la pyramide les **tests IHM** (interaction homme-machine). Ils vont consister à s'assurer de l'ergonomie de l'interface graphique ; concrètement, cela passe par la présentation visuelle et la navigation au sein du produit.

4.3.1. Exemple de réalisation des tests de la pyramide de Cohn dans un projet

Prenons un exemple concret d'enchainement des tests au cours d'un projet, afin de mieux comprendre les niveaux de tests.

L'exigence à réaliser est la suivante : « *En tant que responsable de formation, je veux que vous réalisiez un outil capable de générer à la demande un trombinoscope au format .pdf, afin que je puisse le transmettre au corps enseignant à chaque mise à jour des listes des étudiants* ».

4.3.1.1. Réalisation d'un test unitaire

Nous allons tester la portion de code qui est chargé de l'affichage des étudiants par ordre alphabétique.

Ce projet ayant été réalisé en langage PHP, nous utilisons le framework de test open source PHPUnit[17] qui propose un grand nombre de méthodes d'assertion ; une assertion permet de vérifier une proposition, par exemple : « *le résultat attendu est égal au résultat renvoyé.* ».

Nous avons intégré PHPUnit dans NetBeans, notre IDE (Integrated Develoment Environnement), afin de pouvoir lancer les tests rapidement tout au long de notre processus de développement et éviter de passer les instructions en ligne de commande.

[17] https://phpunit.de/

Voici le code à tester :

```php
<?php
class Utils{
        public static function sortArray($tab){
                sort($tab);
                return $tab;
        }
}
```

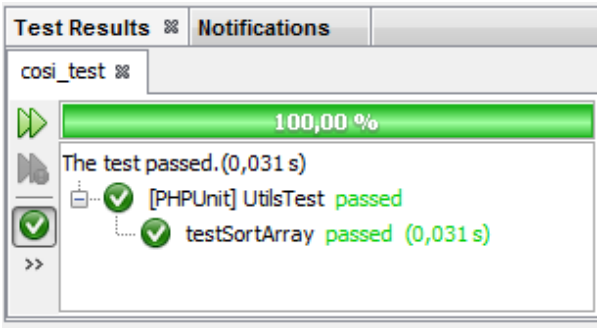
Et le script de test unitaire correspondant :

```php
<?php

class UtilsTest extends PHPUnit_Framework_TestCase {

    public function setUp() {
        require_once '../Utils.php';
    }

    public function testSortArray() {
        $test = array('Durand', 'Dupont', 'Martin');
        $result = Utils::sortArray($test);
        $expected = array('Dupont', 'Durand', 'Martin');
        $this->assertEquals($expected, $result);
    }
}
```

Nous chargeons la bibliothèque de test PHPUnit en ligne 3, puis la classe **`Utils`** que nous voulons tester en ligne 6.

En ligne 13, au travers de la méthode d'assertion **`assertEquals()`**, nous testons le fait que le résultat attendu est bien une suite de noms classés par ordre alphabétique.

Nous lançons ensuite le test dans notre IDE afin de voir le résultat :



Comme nous pouvons le constater, les tests sont souvent plus longs à écrire que le script à vérifier ; toutefois, à chaque modification du code source, nous pouvons aisément rejouer nos tests pour nous assurer qu'il n'y a pas de régression.

4.3.1.2. Réalisation d'un test fonctionnel

Les tests fonctionnels se situent à un niveau plus élevé que les tests unitaires puisqu'ils s'attachent à vérifier le fonctionnement de la couche de présentation, en l'occurrence l'interface web de nos produits. CasperJS est un utilitaire qui permet de manipuler grâce à son API

(interface de programmation) un navigateur Internet en accédant directement au DOM (Document Object Model) de ce dernier. A noter que les tests fonctionnels sont indépendants du langage dans lequel a été écrite l'application web, puisque nous manipulons directement le navigateur.

Dans la pratique, nous écrivons un script de test fonctionnel en Javascript qui sera traité par CasperJS.

Pour reprendre notre exemple précédent, nous allons à présent écrire un test fonctionnel permettant de vérifier la présence du lien permettant de générer un trombinoscope après le choix d'une promotion d'étudiants :

```
1  casper.test.begin('Test présence lien de téléchargement', function(test) {
2      casper.start('http://www.isthia.fr/adminsite/contents/trombinoscope/index.php?promo=2016');
3      casper.waitForSelector(x("//*[contains(text(), \'Télécharger le trombinoscope\')]"),
4          function success() {
5              test.assertExists(x("//*[contains(text(), \'Télécharger le trombinoscope\')]"));
6            },
7          function fail() {
8              test.assertExists(x("//*[contains(text(), \'Télécharger le trombinoscope\')]"));
9      });
10
11     casper.run(function() {test.done();});
12 });
```

En ligne 1 nous lançons la suite de tests et en ligne 5 et 8 nous utilisons la méthode d'assertion **`assertExists()`** permettant de vérifier l'existence d'un élément au sein de la page web, en l'occurrence l'intitulé du lien de téléchargement du trombinoscope.

Nous lançons ce test à l'aide de l'application ASIPAT afin de voir le résultat :

Résultats du test

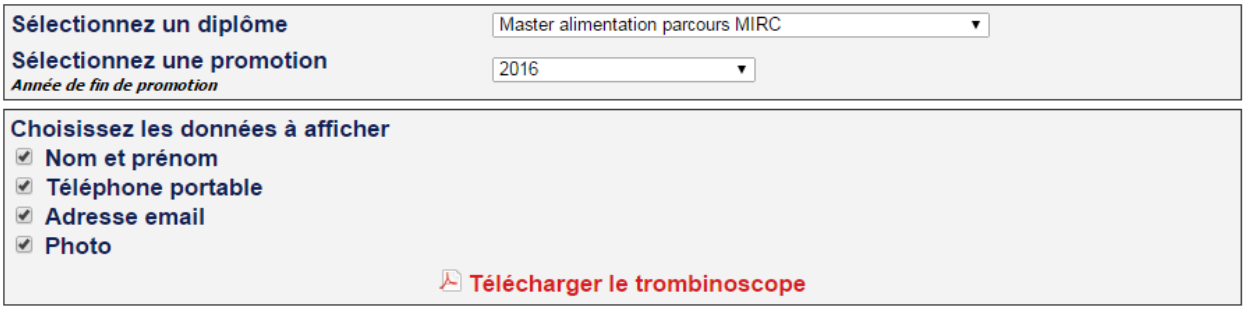
Aucun échec
Nombre de tests: 1
Erreurs: 0
Durée: 3.296 secondes

4.3.1.3. Réalisation d'un test IHM

En plus de son API de test, CasperJS permet aussi d'écrire des scripts permettant de définir et d'ordonner des étapes de navigation au sein d'un site web.

Nous pouvons générer des captures d'écran afin de consulter le rendu du site web tout au long des étapes de navigation. Ces captures peuvent ensuite être présentées au PO pour attester de la conformité de l'interface graphique des produits avec ses exigences.

Concernant notre exemple, nous pouvons générer une capture d'écran après l'exécution du test fonctionnel précédent et ainsi constater le rendu visuel de la page web :



4.3.2. Choisir de développer une application autour des tests fonctionnels

A l'heure actuelle, nous n'avons pas encore mis en œuvre de campagne de tests unitaire dans nos développements, même si l'environnement de tests unitaires que nous avons présenté est prêt. En effet, le CMS propriétaire CosiX à partir duquel nous développons tous nos produit a été conçu il y a huit ans par mes soins, époque à laquelle je n'étais pas familier des tests unitaires. Toutefois, à compter du second semestre 2015, nous allons progressivement tester le code source des nouveaux développements que nous entamerons.

Pourquoi choisir de mettre en œuvre des tests fonctionnels dans nos projets agiles ?

Suite à un entretien avec Jean-Michel Inglebert, enseignant-chercheur à l'IUT de Blagnac et spécialiste des méthodes agiles, je me suis rendu compte que le lien entre les tests unitaires et les exigences du PO est hors sujet, ce qui n'est pas le cas des tests fonctionnels. **Il est ainsi tout à fait logique et pertinent d'associer aux exigences tous les tests fonctionnels réalisés**. Dans un projet agile, le PO est en effet intéressé par le résultat des tests permettant de vérifier le fonctionnement de son application, plutôt que par la qualité du code source de cette dernière. Pour finir, le framework de tests fonctionnels CasperJS permet de générer des captures d'écran suite aux tests réalisés (en succès ou en échec) que nous pouvons présenter à notre PO et qui sont immédiatement compréhensibles par ce dernier, même s'il ne possède pas de compétences en informatique.

C'est pour ces différentes raisons que l'application ASIPAT a été développée autour des tests fonctionnels.

Nous allons maintenant présenter ASIPAT plus en détail et nous focaliserons sur l'application locale qui présente le plus d'intérêt en termes techniques et en ce qui concerne les processus métiers mis en œuvre. Au moment de l'écriture de ces lignes, l'application distante est en cours de développement, nous présenterons toutefois le framework qui servira de socle technique à cette dernière.

4.4. Présentation du « bac à sable », le cœur de l'application locale

Le « *bac à sable* » est le cœur de l'application locale d'ASIPAT. Il permet de jouer et d'enregistrer des scripts de tests fonctionnels ou de navigation, puis de les rattacher éventuellement à une exigence d'un projet agile.

> **Remarque :** en plus des tests fonctionnels, nous avons choisi de permettre de rattacher un script de navigation à une exigence ; cela peut en effet présenter un intérêt pour le PO. Par exemple, nous pouvons réaliser un script de navigation permettant de simuler l'affichage d'une page web selon plusieurs définitions d'écran afin de vérifier son rendu en fonction de plusieurs types de terminaux ; dans ce cas-là, ce n'est pas un test fonctionnel, mais cela peut tout à fait être pertinent de le rattacher à une exigence du PO.

L'interface s'articule en trois sections situées dans une page unique :

- La section de lancement d'un script.
- L'affichage du résultat de l'exécution du script de test ou de navigation.
- L'enregistrement et le rattachement du script à une exigence.

4.4.1. Lancement d'un script de test fonctionnel ou de navigation

Figure 21 : Aperçu de l'interface de lancement d'un script de test fonctionnel ou de navigation dans l'application ASIPAT.

Cette section permet de lancer un script de test ou de navigation qui sera traité par CasperJS. Afin de faciliter la lecture et la modification d'un script directement dans ASIPAT, nous avons mis en place un éditeur de code à l'aide du **plugin CodeMirror**[18] qui permet d'assurer trois fonctions :

- La coloration syntaxique du code.
- Une aide à l'écriture de fonctions JavaScript grâce à un mécanisme d'autocomplétion.
- La détection des erreurs de syntaxe.

[18] http://codemirror.net/

Par ailleurs, l'écriture de tests fonctionnels étant souvent longue, nous utilisons l'**extension Resurrectio**[19] pour Google Chrome qui permet d'enregistrer une séquence d'action dans le navigateur et de produire le code JavaScript correspondant pour CasperJS. Cela offre un gain de temps important, puisqu'il suffit ensuite de recopier ce code dans l'éditeur d'ASIPAT et de lancer le script.

4.4.2. Affichage du résultat de l'exécution du script de test ou de navigation

Pour permettre le lancement de CasperJS via l'application ASIPAT, nous employons la fonction `exec()` de PHP. Cette dernière permet l'exécution d'un programme externe, puis fournit en valeur de retour le résultat de la commande. On affiche ainsi ce résultat dans une console de sortie. Dans le cas d'un script de test fonctionnel, on demande à CasperJS de générer un fichier XML contenant le détail des logs des résultats du test.
De plus, ASIPAT est capable de générer une capture d'écran qui sera affichée automatiquement lors de l'échec d'un test. Cela permet au développeur de connaitre la raison de cet échec.

[19] https://github.com/ebrehault/resurrectio

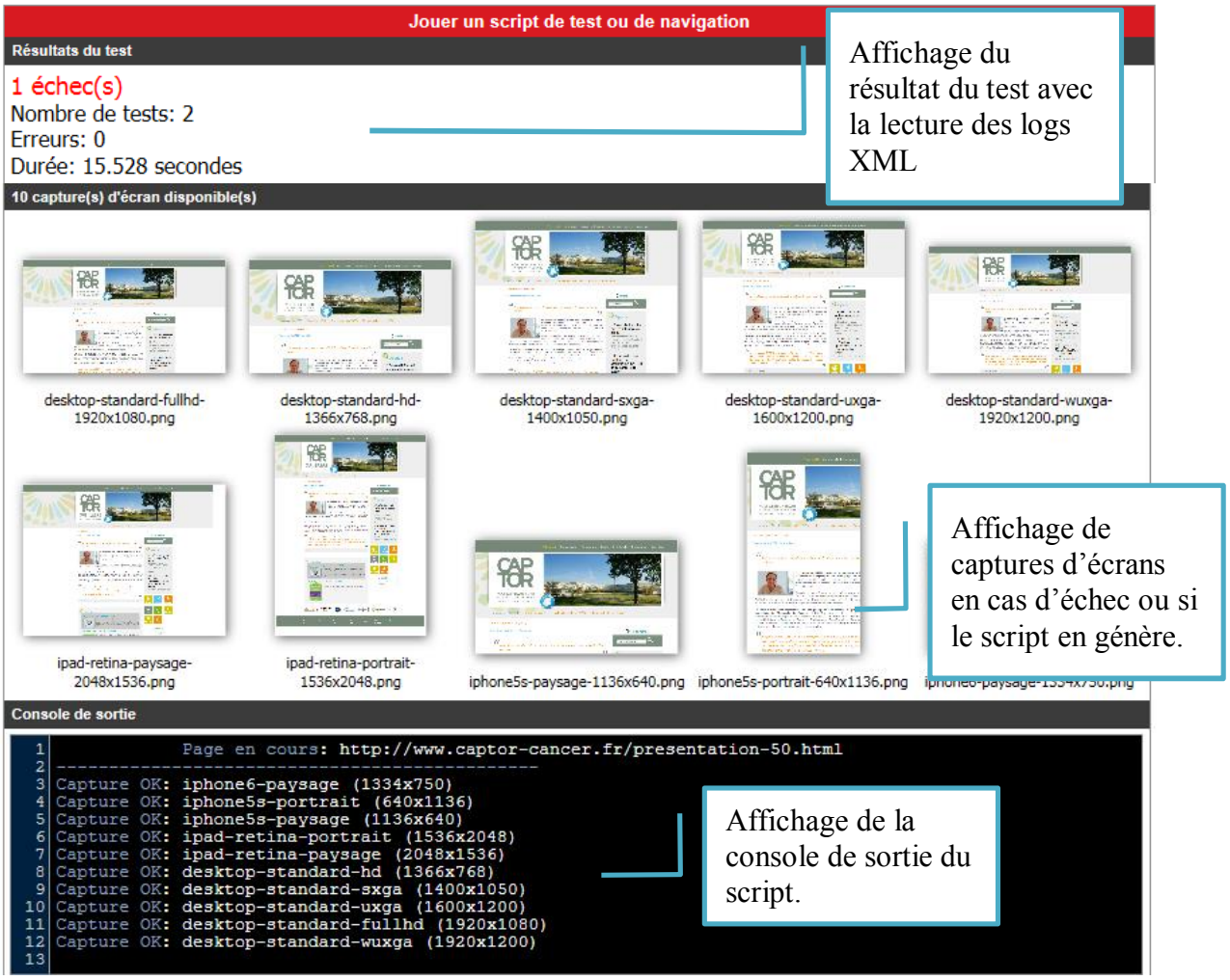


Figure 22 : Aperçu de l'affichage du résultat de l'exécution d'un script de test ou de navigation dans l'application ASIPAT.

4.4.3. Enregistrement et rattachement d'un script à une exigence

L'utilisateur de l'application peut choisir d'enregistrer un script en base de données afin de le conserver pour le jouer à nouveau quand il le jugera nécessaire. Il est aussi possible de modifier un script déjà enregistré.

L'utilisateur a aussi la possibilité de rattacher un script de test ou de navigation à une exigence référencée dans l'application ASIPAT distante. Nous avons développé pour cela un service web capable de mettre à jour la base de données de cette dernière. Le service web présente ainsi deux intérêts :

- Permettre le rattachement d'un script à une exigence sans avoir à se connecter à l'interface d'administration de l'application distante. Cela simplifie la procédure pour le testeur.

- Offrir la possibilité au PO de consulter en direct l'état d'avancement des tests fonctionnels.

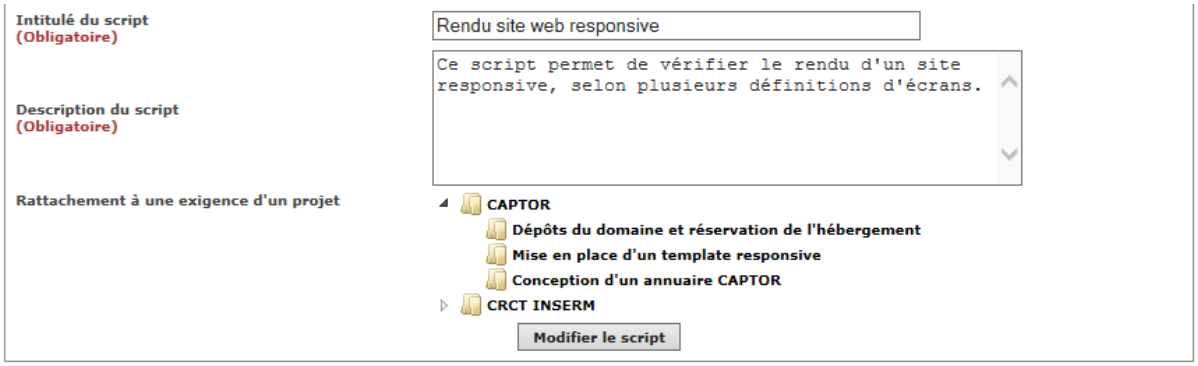


Figure 23 : Aperçu de l'interface d'enregistrement et de rattachement d'un script à une exigence.

Dans l'exemple ci-dessus, l'arborescence représente les exigences des deux projets pilotes enregistrées dans l'application distante. Ces informations sont affichées en local en consommant le service web d'ASIPAT distant. Le testeur peut rattacher le script intitulé « *Rendu site web responsive* » à l'exigence de son choix, en cliquant sur cette dernière dans l'arborescence.

4.5. Présentation des principaux processus métiers

A l'aide de diagrammes d'activité, nous allons présenter les deux principaux processus mis en œuvre dans l'application locale d'ASIPAT.

Le premier concerne le lancement et l'enregistrement d'un script de test fonctionnel ou de navigation et le second, l'association d'un script à une exigence.

4.5.1. Jouer et enregistrer un script de test fonctionnel ou de navigation

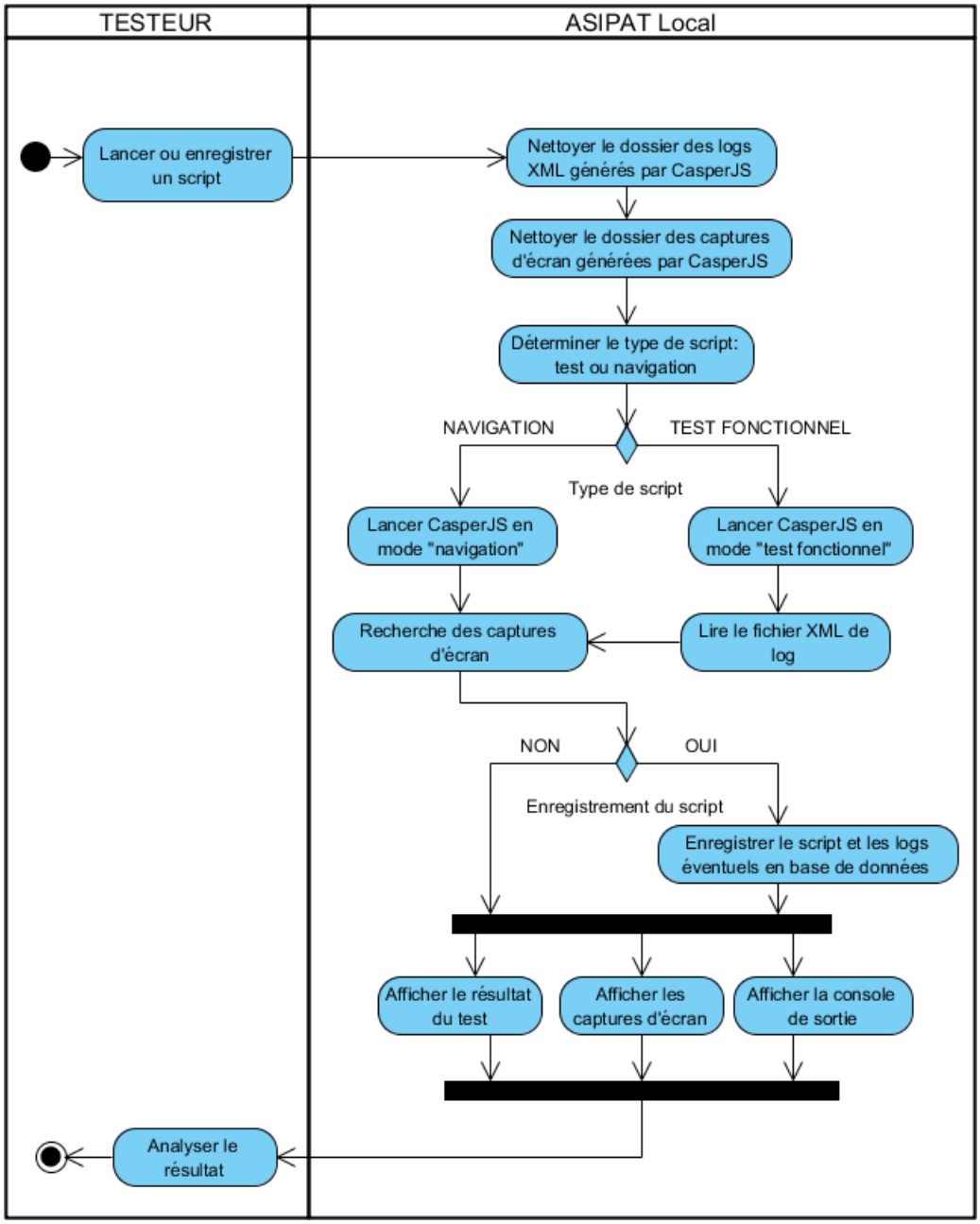


Figure 24 : Diagramme d'activité «Jouer et enregistrer un script de test fonctionnel ou de navigation dans l'application ASIPAT».

Nous remarquons avec ce diagramme d'activité que lorsque l'utilisateur décide d'enregistrer un script, ce dernier est tout d'abord lancé dans CasperJS afin de pouvoir récupérer les logs et les enregistrer, dans le cas d'un test fonctionnel. Cela permet aussi au testeur de vérifier une nouvelle fois le résultat du test.

Nous voyons aussi que les fichiers des logs et les captures d'écrans ne sont pas conservés sur le serveur. Cela ne présente aucun intérêt de conserver les captures d'écran, puisque la page web

évaluée par le script peut évoluer durant le projet. Il est donc nécessaire d'en générer de nouvelles à chaque lancement du script. Les logs, quant à eux, sont conservés en base de données afin de pouvoir informer le PO du résultat d'un test fonctionnel lors du rattachement à une exigence. A chaque modification d'un script de test fonctionnel déjà enregistré, **l'application locale met à jour le rattachement avec l'exigence distante** s'il en existait un auparavant ; cela permet d'éviter un décalage dans le cas où le résultat d'un test change entre temps.

4.5.2. Associer un script de test fonctionnel ou de navigation à une exigence

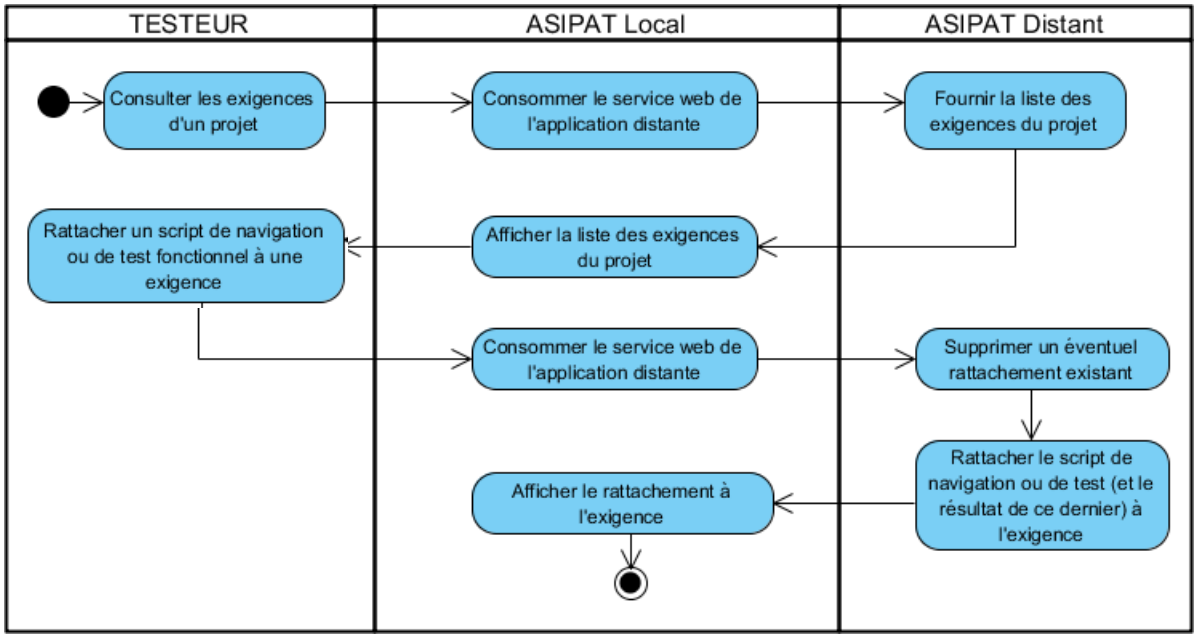


Figure 25 : Diagramme d'activité «Associer un script de test fonctionnel ou de navigation à une exigence dans l'application ASIPAT».

Comme nous l'avons déjà expliqué, les exigences des projets sont enregistrées et gérées dans l'application distante. L'application locale n'est chargée que de la gestion des scripts, elle consomme ainsi le service web de l'application distante à de nombreuses reprises, notamment pour afficher les exigences.

Nous allons maintenant présenter les principaux choix techniques que nous avons adoptés pour concevoir ASIPAT.

4.6. Choix techniques

4.6.1. Le CMS propriétaire CosiX comme socle technique

J'ai choisi de développer ASIPAT en me servant de la base technique du CMS CosiX pour trois raisons :

- **Dans une visée d'efficience** : Ayant conçu l'intégralité du CMS, nous avons acquis des automatismes de développement avec ce dernier nous permettant d'implémenter plus rapidement certaines fonctionnalités comme la mise en place des traitements asynchrones.
- **Dans une visée de sécurité** : CosiX étant une solution propriétaire et l'architecture et le code source n'étant pas diffusé auprès du grand public, nous bénéficions du principe de sécurité par l'obscurité. Cela contribue à diminuer le risque de piratage.
- **Dans une visée de rapidité** : cela nous évite de redévelopper une interface graphique et des fonctionnalités de base comme la gestion des comptes utilisateurs et l'authentification à l'application.

Nous voyons aussi deux inconvénients dans l'utilisation de Cosix :

- Le principal inconvénient vient du fait que je suis le seul développeur à connaitre l'architecture de CosiX et son code source. La solution est ainsi potentiellement difficile à maintenir et à faire évoluer par d'autres développeurs.
- ASIPAT fait partie intégrante de CosiX et il est délicat d'en faire un module stand-alone réutilisable dans un autre CMS.

Sachant qu'à moyen terme, je vais être le seul développeur à utiliser ASIPAT, les avantages de la solution CosiX ont pris le pas sur les inconvénients.

4.6.2. L'utilitaire CasperJS pour les tests fonctionnels

Nous avons déjà expliqué l'intérêt de CasperJS qui est un utilitaire permettant d'écrire des tests fonctionnels grâce à son API très riche, mais aussi de réaliser des scripts permettant de simuler une navigation dans des pages web.

Il existe des framework concurrents, le principal étant Selenium. Nous avons élaboré un tableau comparatif des deux solutions en nous basant sur notre expérience personnelle et sur le tutoriel Selenium de Thomas (2013).

	CasperJS	Selenium	Avantage pour
Installation	Très simple mais la configuration est un peu plus complexe.	Très simple, il suffit d'installer une extension dans FireFox.	**Selenium**
Utilisation	Nécessite de lancer des lignes de commande, sauf dans le cas de notre application ASIPAT.	Très simple, il suffit de passer par l'interface de l'extension Firefox de Selenium.	**Ex aequo** (avec l'utilisation d'ASIPAT)
Simplicité d'écriture des tests	Nécessité de connaitre le JavaScript. Possibilité de se servir de l'extension Chrome Resurrectio pour générer des scripts.	Aucune écriture de tests, c'est l'enregistreur d'action de Selenium qui s'en charge. On peut aussi insérer des commandes manuellement.	**Selenium**
Possibilités de tests	Très importantes grâce à son API de test. Possibilité d'élaborer des tests fonctionnels très pointus.	Le niveau d'abstraction est élevé, la prise en main est plus rapide, mais gare aux faux positifs dus à un enchainement de commandes mal élaboré.	**CasperJS**
Communauté	Des tutoriels de qualité essentiellement en anglais. Difficile d'obtenir de l'aide sur les forums.	Plus de tutoriels que pour CasperJS, communauté plus importante.	**Selenium**
Intégration	Possibilité d'intégrer CasperJS dans un CMS comme CosiX.	Possibilité d'intégrer Selenium dans un IDE comme Eclipse.	**Ex aequo** (dépend de l'usage)

Tableau XI : Comparatif entre les outils de tests fonctionnels CasperJS et Selenium.

Notre volonté de développer une application autour des tests fonctionnels et le fait que CasperJS puisse être totalement intégré dans notre CMS CosiX a été le facteur déterminant à son adoption.

4.6.3. Les services web pour les échanges de données

Un service web permet l'échange de données dans un environnement distribué, c'est-à-dire dans lequel les applications ne sont pas sur les mêmes machines et peuvent être écrites dans des langages de programmation différents.

Dans le cadre de ce mémoire, la problématique était de faire communiquer l'application locale et distante d'ASIPAT avec les contraintes suivantes :

- Un échange de données rapide.

- Une solution simple à implémenter et facile à appréhender.

Afin de prendre en compte ces contraintes, nous avons choisi de nous appuyer sur le protocole SOAP. Le langage PHP que nous pratiquons possède une extension dédiée très simple d'utilisation.

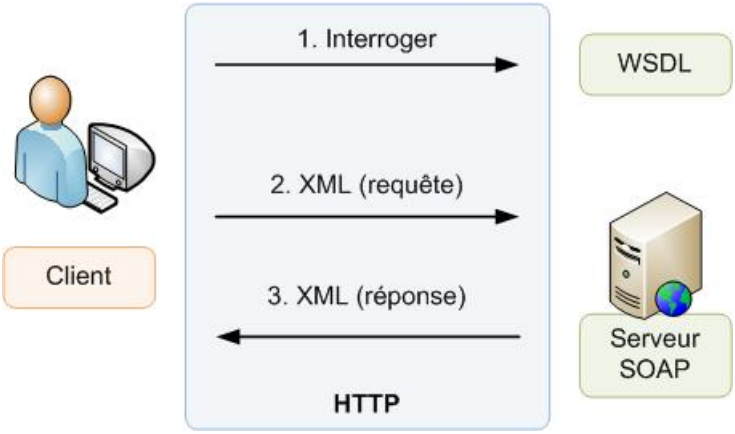


Figure 26 : Principe d'échange des données via le protocole SOAP (Simple Object Access Protocol) (Poiron, 2010).

Par ailleurs, dans le cas d'un dialogue entre deux applications écrites en PHP, il n'est pas nécessaire de concevoir un fichier de description de service WSDL (Web Services Description Language) : cela nous a permis de simplifier la mise en place du service web. A moyen terme, il n'est pas prévu en effet de faire dialoguer ASIPAT avec une application écrite dans un autre langage.

La critique principale du protocole SOAP concerne le format XML d'échange des données qui peut ralentir le temps d'acheminement des messages (Poiron, 2010). En pratique, j'ai été au contraire très surpris par la rapidité d'échange des données, je n'ai pas remarqué de ralentissements dans l'affichage des pages qui consomment le serveur web de l'application distante. J'ai pu mesurer un temps moyen d'affichage du « *bac à sable* » d'ASIPAT de cinq-cent-vingt millisecondes (moyenne sur vingt rechargements de page).

4.6.4. Le framework jQuery Mobile pour l'affichage adaptatif multi-écrans

Nous le constatons avec l'ensemble de nos clients, il est indispensable de proposer aujourd'hui une solution web permettant une consultation des pages adaptée à tous types d'écrans : smartphones, tablettes, moniteurs d'ordinateurs.

C'est pour cette raison que nous avons choisi d'utiliser le framework jQuery Mobile[20] pour l'interface dédiée au PO de l'application distante d'ASIPAT. jQuery Mobile permet de concevoir un site web adaptatif (couramment appelé « *responsive* »).
L'objectif est de permettre au PO de se connecter à ASIPAT depuis n'importe où. En effet, selon Médiamétrie (2014), 54% des internautes français déclarent surfer sur les pages web avec leur smartphone.
jQuery Mobile est intégré au CMS CosiX, et l'administration de ce dernier permettra au Scrum-master de renseigner les projets, les exigences, les sprints, les tâches et la gestion des comptes des PO pour l'authentification dans ASIPAT (voir page 76).
Nous allons maintenant tenter de voir les atouts qu'offre ASIPAT pour CosiWeb.

4.7. Atouts de l'application

ASIPAT a été conçu pour répondre à un besoin que nous avions constaté : les tests fonctionnels réalisés jusqu'à présent étaient manuels et incomplets, ce qui entrainait régulièrement une constatation tardive de bogues. L'autre atout de l'application est de faciliter le suivi d'un projet agile pour le PO et de lui permettre de constater la couverture de ses exigences par des tests fonctionnels.
En reprenant le tableau en page 74 des besoins de CosiWeb non comblés par Scrum, nous constatons qu'ASIPAT permet de combler les dysfonctionnements liés aux tests fonctionnels. Pour ce qui est de la réalisation de tests unitaires, nous allons les intégrer progressivement dans nos pratiques.

5. Résumé

Nous avons fait état dans ce chapitre de l'expérimentation que nous avons faite de l'agilité avec la méthode Scrum, au travers de deux projets pilotes. Nous avons également présenté l'application ASIPAT que nous avons conçue afin de répondre notamment aux besoins constatés en matière de tests fonctionnels.
Maintenant que nous avons présenté la façon dont nous avons mis en pratique l'agilité tant en matière de gestion de projet que d'ingénierie logicielle, nous disposons d'un certain recul nous permettant d'en faire une analyse critique.

[20] http://jquerymobile.com/

Analyse des limites de l'agilité et préconisations

Dans ce chapitre, nous allons mettre en relief certaines limites de la démarche agile que nous avons pu constater. Nous chercherons ensuite à voir quelle est la prédisposition de nos clients à l'agilité et nous terminerons par des préconisations que nous pourrions faire à une entreprise qui voudrait se lancer dans cette démarche.

1. Limites de la démarche agile

1.1. Les contraintes liées à la contractualisation

Comme nous l'avons vu précédemment, dans le cadre d'une méthode agile, le projet peut se terminer même si toutes les exigences du client n'ont pas été développées, en raison d'une enveloppe budgétaire insuffisante ou pour des questions de délais trop courts. C'est une situation impossible dans le cadre de contrats au forfait que nous signons notamment lors des marchés publics. Est-ce une problématique propre à l'entreprise CosiWeb? Assurément non. Citons ainsi Beaugrand (2013) qui met en garde sur les « *pièges de la contractualisation agile [...] qui peut être un chemin semé d'embûches* » et qui constate que « *le manifeste agile lui-même minore l'importance du contrat* ». Lors de « *l'agile tour Toulouse* », j'ai par ailleurs soulevé cette difficulté à la mise en œuvre d'une méthode agile dans le cadre de réponse à des marchés et je n'ai pas eu de réponse faisant consensus.

Il y a très peu d'allusions aux contrats dans la littérature agile et lorsque c'est le cas, les réponses apportées sont pour le moins surprenantes. Aubry, cité dans l'ouvrage de Messager (2010), n'hésite pas à déclarer que « *la contractualisation des relations entre le client et l'équipe de réalisation n'est pas obligatoire ni même recommandée par les méthodes agiles* » ! Il propose, si la contractualisation est indispensable, l'élaboration d'un appel d'offre agile. Hélas, les concepteurs d'appels d'offre ne semblent pas sensibilisés à cette question ; je n'ai d'ailleurs jamais eu connaissance d'un tel document depuis le début de notre activité.

Nous allons maintenant présenter de manière critique la façon dont l'agilité peut parfois être présentée comme l'alternative incontournable et absolue à toute autre méthode de gestion de projet. Pour ce faire, nous nous sommes appuyés sur la conférence de Meyer (2013) lors des journées nationales du développement logiciel.

1.2. L'agilité, un courant dogmatique ?

Rappelons la définition du dogmatisme par le Larousse : « *qui a des opinions bien arrêtées, qui les considère comme des vérités absolues, et les exprime d'une manière péremptoire, autoritaire, catégorique.* »

Meyer (2013) cite un article de Forbes dans lequel l'auteur, Denning, explique que les personnes n'adhérant pas à l'agilité acceptent la médiocrité, la bureaucratie et sont incompétents ; il ajoute « *when the culture doesn't fit agile, the solution is not to reject agile. The solution is to change the organizational culture* ». En d'autres termes, il ne peut y avoir d'autre solution que l'agilité !

On retrouve dans la littérature agile ce clivage entre ceux qui adoptent les méthodes agiles et les autres ; ainsi Schwaber et Sutherland (2012), les créateurs de Scrum, déclarent « *you have been ill served by the software industry for 40 years […]. We want to restore the partnership* ».

Un autre aspect marquant des ouvrages relatifs à l'agilité est la récurrence des termes appartenant au champ lexical des religions. Prenons l'exemple de la préface de l'ouvrage de Messager (2010), écrite par Tabaka qui se revendique « *évangéliste* » du mouvement agile et qui parle de sa « *croyance profonde* » dans les valeurs qu'il prodigue ! La réunion de février 2001 ayant donné lieu à la publication du manifeste agile (voir page 41) est présentée comme un événement quasi-mystique dans la plupart des ouvrages, rappelons que cela avait été en premier lieu une occasion de faire du ski, se relaxer et manger (Highsmith, 2001) !

J'ai d'autre part pu constater tout comme Meyer (2013) que de nombreux ouvrages de la littérature agile se basent sur l'enquête de 2002 du Standish Group pour démontrer que les entreprises qui n'ont pas recours à l'agilité voient leurs projets menés à l'échec dans 74% des cas. Notons que les critères sur lesquels ils se basent pour considérer un projet comme un succès, sont le respect du budget, des délais et de la conformité du produit aux exigences du client, et rappelons que dans le cadre d'une démarche agile, un projet peut être livré dans les temps mais sans garantir la réalisation de toutes les fonctionnalités, ou avec toutes les demandes réalisées mais sans garantir les délais (Beck & Andres, 2004) ! A notre connaissance, nous n'avons d'ailleurs trouvé aucune statistique basée sur les mêmes critères permettant de démontrer le succès des projets menés selon une démarche agile…

Par ailleurs, Meyer (2013) critique les partisans de l'agilité qui donnent fréquemment des preuves de ses apports hypothétiques par l'anecdote plutôt que par la preuve. Il cite Cohn, un des fondateurs de l'agilité, qui tente de justifier une des quatre valeurs du manifeste : « *des logiciels opérationnels plus qu'une documentation exhaustive* », en donnant l'anecdote de sa secrétaire lui ayant dit que la salle de séminaire était réservée, ce dernier ayant compris qu'elle l'était pour lui

alors qu'elle était en fait déjà prise. Il argumente longuement pour tirer la conclusion qu'écrire de la documentation est source d'ambiguïté.
Pour terminer, nous pourrions aussi souligner le fait que les dix-sept experts à l'origine du mouvement agile sont d'autant plus élogieux en la matière qu'ils vendent leurs services en tant que consultants agiles et des ouvrages sur le sujet.

2. Nos clients sont-ils prêts pour l'agilité ?

2.1. Elaboration d'une enquête par questionnaire

En novembre 2014, nous avons élaboré et administré auprès d'une partie de nos clients un questionnaire dont l'objectif était d'évaluer leur prédisposition à l'agilité. Afin de déterminer l'échantillon des clients à sonder, nous nous sommes basés sur les critères suivants :

- Nous avons mené avec le client un projet terminé.
- Le projet doit représenter notre cœur de métier, il doit donc consister en la conception d'un site web avec des fonctionnalités spécifiques à développer.
- Le projet ne doit pas avoir été mené dans l'agilité.

Nous avons administré le questionnaire comportant onze questions fermées à un échantillon de vingt-huit personnes (nos principaux interlocuteurs sur les projets) et vingt-deux ont répondu (confère annexes page 103), soit près de huit clients sur dix.

2.2. Prédisposition de nos clients à l'agilité

En question préliminaire, nous avons demandé à nos clients de classer par ordre d'importance les trois éléments du triangle de la performance ; arrive en premier lieu la conformité du produit aux exigences, puis le respect du budget et enfin le respect des délais.
Nous avons ensuite abordé la question de la contractualisation qui est selon moi la principale problématique des méthodes agiles. Contrairement à une méthode classique où les principaux éléments contractuels sont intangibles (périmètre fonctionnel défini dès la proposition commerciale initiale, prix forfaitaire, durée bornée dans le temps (Beaugrand & Belin, 2013)), les méthodes agiles proposent une alternative dans laquelle le projet peut prendre fin à chaque fin d'itération pour trois raisons:

- Le client décide que le produit répond à tous ses besoins.
- L'enveloppe budgétaire de ce dernier est consommée.
- Le délai de mise en œuvre du produit est atteint.

Quelles qu'en soient les raisons, le produit sera totalement opérationnel, même si dans certains cas il ne répondra pas à toutes les exigences du client.

Ainsi, nous avons posé des questions tournant autour de l'adaptation aux besoins que proposent les méthodes agiles :

Durant le développement de votre site web	Oui	Non	Je ne sais pas
Aimeriez-vous pouvoir ajouter de nouvelles fonctionnalités non prévues initialement ?	100%	0%	0%
Aimeriez-vous avoir la possibilité de modifier les fonctionnalités initiales, voire de les supprimer si elles ne correspondent plus à votre besoin ?	95%	5%	0%

Tableau XII : Volonté des clients de CosiWeb de bénéficier d'un produit adapté à leurs besoins tout au long du projet.

Ces résultats montrent que nos clients sont très intéressés par cette dimension de l'agilité. Mais si nous posons ensuite la question suivante, les réponses sont sans équivoque :

Au moment de signer le contrat	Oui	Non	Je ne sais pas
Accepteriez-vous de contractualiser pour un site web qui possèdera toutes les fonctionnalités souhaitées, mais ne pas avoir d'échéance ferme avec un budget qui pourra être révisé à intervalle régulier ?	18%	77%	5%

Tableau XIII : Volonté des clients de CosiWeb de s'engager dans une contractualisation agile.

Nous touchons du doigt la difficulté principale à adopter une méthode agile dans le contexte commercial dans lequel évolue notre entreprise, nos clients sont très intéressés par les principes proposés par l'agilité mais sans les concessions que cela peut impliquer. De plus, une dernière série de questions permet de mettre en évidence le fait que nos clients sont très attachés à ce que le produit réponde à toutes leurs exigences, même celles qui sont secondaires :

En sachant que les fonctionnalités les plus importantes seront développées en priorité.	Oui	Non	Je ne sais pas
Accepteriez-vous que le site web ne possède pas toutes les fonctionnalités demandées mais juste celles qui sont essentielles ?	23%	59%	18%

Tableau XIV : Volonté des clients de CosiWeb à voir toutes leurs exigences développées dans le produit final.

2.3. Bilan de l'enquête

Nous pouvons constater que **nos clients sont réceptifs aux principes de l'agilité**, mais que l'adoption d'une méthode complète (et non pas d'une sélection de pratiques comme nous l'avons

fait dans le cadre de ce mémoire) peut s'avérer être très complexe car elle implique des concessions que nos clients ne sont pas prêts à faire. Nous avons par ailleurs abordé les contraintes liées à la contractualisation agile.

Nous avons ainsi vu que l'on peut mettre en question l'argumentation souvent répandue selon laquelle l'agilité serait la solution à tous les maux de l'entreprise. Avec l'expérience que nous avons acquise dans sa mise en pratique, nous allons proposer quelques préconisations à une entreprise qui souhaiterait se lancer dans cette démarche.

3. Préconisations

Après avoir expérimenté les pratiques agiles tant en matière de gestion de projet que d'ingénierie logicielle, et au vu des limites que nous pouvons constater concernant l'agilité, nous allons tenter de lister les préconisations que nous pourrions donner à une entreprise qui voudrait se lancer dans cette démarche.

3.1. Eviter de se lancer dans l'agilité pour de mauvaises raisons

« J'ai entendu parler de l'agilité et je suis curieux de l'expérimenter dans mon entreprise. »

Contrairement à ce que soutiennent de nombreux agilistes comme Schwaber et Sutherland (2012), on ne peut pas mettre de côté quarante ans de gestion de projet classique. Si l'entreprise ne rencontre pas de dysfonctionnements présentant un risque pour son activité, on peut se poser la question de l'intérêt de se lancer dans une nouvelle démarche de gestion de projet.

« Je dois répondre à un marché dans lequel il est indiqué qu'une méthode agile serait appréciée. »

Mettre en œuvre des pratiques agiles requiert du temps et de la méthodologie, ce serait faire preuve d'une certaine inconscience de franchir le pas sans mesurer la conséquence des changements que cela peut induire sur l'entreprise et les personnes qui l'animent. L'entreprise est un organisme « *vivant* » qui a tendance à opposer de la résistance aux changements, il ne faut pas la négliger au moment de passer à l'agilité.

« Je rencontre des dysfonctionnements dans mon entreprise et j'ai entendu dire que l'agilité était la solution idéale. »

D'une part, il est indispensable de déterminer la nature de ces dysfonctionnements et de mesurer le risque qu'ils font courir à l'entreprise ; d'autre part, il est primordial de s'assurer que les pratiques et les outils proposés par les méthodes agiles peuvent y répondre. Nous proposons dans

ce mémoire une méthode de recensement, de classification et d'évaluation des dysfonctionnements existants (voir page 21).

3.2. Démarrer l'agilité par étapes

Nous préconisons de débuter une démarche agile par les outils avant les pratiques. On peut en effet élaborer un **product-backlog** en recueillant les exigences de son client, puis se servir d'un **task board** pour suivre l'avancée du projet et enfin mettre en place une **roadmap** pour identifier les étapes clefs du projet. Les pratiques propres aux méthodes agiles peuvent arriver ensuite, cela permet d'une part d'aborder l'agilité avec plus de sérénité et d'autre part de permettre à l'entreprise de se préparer aux changements organisationnels que cela implique.

3.3. Proposer des outils simples d'utilisation

Un des facteurs de succès d'un projet agile est la collaboration du PO avec l'équipe de développement. Cette collaboration passe par l'utilisation d'outils qui doivent être faciles à appréhender afin d'être utilisés de façon régulière. Un task board avec des post-it est simple à appréhender, et si l'on développe un outil plus sophistiqué comme l'est notre application ASIPAT (voir page 75), il faut que ce dernier soit utilisable et compréhensible pour le PO avec quelques explications mais sans formation préalable.

3.4. Choisir les projets qui peuvent être menés dans l'agilité

Il est illusoire de penser que tous les projets peuvent être menés dans l'agilité. Nous pensons qu'il faut être vigilant aux deux points suivants pour qu'un projet puisse être candidat:

- Le PO doit montrer sa volonté de se lancer dans la démarche. Contraindre un client à adopter les principes des méthodes agiles serait certainement contre-productif.
- Si le PO est le payeur, ce dernier rentrera en tension avec l'entreprise dans le cas où toutes les exigences prévues dans un sprint n'auront pas pu être honorées. Et si ce dernier perd confiance en l'équipe de développement, la démarche agile risque l'échec.

3.5. Savoir faire les bons compromis

Beck (2004) promet la fonctionnalité ou la date de livraison. Hors, il est commercialement très difficile d'expliquer à un PO que l'on ne peut pas garantir la satisfaction de toutes ses exigences. L'enquête que nous avons menée (voir page 95) montre d'ailleurs que ces derniers accordent plus d'importance à ce point qu'au respect du budget ou des délais. Nous recommandons donc

de ne pas transiger sur la réalisation de toutes les exigences mais de faire comprendre au PO que certaines ne pourront éventuellement pas être réalisées à temps ; ce dernier l'acceptera d'autant plus qu'il sera impliqué dans la conception de son produit et comprendra les contraintes de l'équipe. Nous pensons pour finir que contrairement à ce qu'affirment certains auteurs comme Messager (2010) (voir page 51), il est imprudent d'affirmer que tendre à respecter les contraintes de qualité, de délais et du budget d'un projet est la mesure du succès des méthodes classiques uniquement et que les méthodes agiles doivent seulement viser l'apport de valeur ajouté au client.

3.6. Maintenir la communication avec son client et dans son équipe à tout prix

Cette dernière recommandation nous semble la plus importante. La communication avec le client est indispensable au succès d'un projet. Dans les projets agiles que nous avons menés, nous n'avons jamais fait l'économie des revues de sprint avec nos PO. Par extension, les mêlées quotidiennes sont le ciment de l'équipe car elles permettent d'être au fait de l'avancée des travaux et de pouvoir résoudre les problèmes au plus tôt.

Conclusion

Notre problématique était de voir si la mise en œuvre de pratiques agiles pouvait permettre de combler les besoins de l'entreprise CosiWeb en matière de gestion de projet.
Nous avons dégagé quatre grands besoins suite à la mise en œuvre de la méthode d'analyse et de gestion des risques Méhari :

- Bénéficier d'une plus grande adaptabilité avec le client en proposant un produit davantage conforme à ses exigences.
- Bénéficier d'une meilleure visibilité sur le projet en suivant plus précisément son état d'avancement.
- Réduire le risque d'échec du projet qui peut être dû à un manque de communication avec le client (l'effet tunnel) ou une sortie du périmètre du triangle de la performance (la qualité du produit, le budget et les délais du projet).
- Enfin, apporter de la qualité logicielle en réalisant un produit conforme aux exigences parfois mouvantes du client lors d'un projet et en améliorant la fiabilité et la maintenabilité de ce dernier à l'aide des tests unitaires et fonctionnels.

La mise en œuvre des pratiques de la méthode agile Scrum nous a permis d'apporter des solutions concrètes aux trois premiers points, grâce notamment au développement itératif et incrémental qui permet d'ajuster régulièrement le produit aux besoins du client, mais aussi en favorisant la communication au sein de l'équipe et en proposant des outils simples à élaborer et permettant un suivi efficace des projets.
Concernant le dernier point, l'application ASIPAT que nous avons conçue nous permettra de rationaliser l'utilisation de tests fonctionnels lors de nos prochains projets menés dans l'agilité. Notons qu'ASIPAT est destiné à prendre peu à peu le pas sur les outils agiles que nous avons conçus initialement à l'aide d'Excel mais pour lesquels il manque une dimension collaborative et la rapidité d'échange que peut proposer une plateforme web (nous pensons notamment au product-backlog qui recense les exigences de nos clients ou encore au burndown chart permettant de suivre l'avancement du projet). Le besoin qu'il reste à combler concerne la pratique des tests unitaires encore trop faible actuellement et que nous allons mettre en œuvre de façon systématique à compter du second semestre 2015.
Au-delà des apports dont nous avons pu bénéficier avec ces pratiques et outils, quelques questionnements restent en suspens. En effet, comme le souligne Meyer (2013), une méthode agile promet la réalisation de toutes les exigences ou la date de

livraison du projet ; cette logique se tient quand on considère qu'une fonctionnalité pourra être remodelée en fonction de l'évolution des besoins du propriétaire du produit. Toutefois, nous avons pu montrer grâce à l'enquête menée auprès de nos clients que ces derniers ne veulent pas transiger sur la satisfaction de toutes leurs exigences dans des délais et pour un budget définis au départ. C'est d'autant plus vrai dans le cas de nos réponses à des marchés publics, un dépassement de la date de livraison ou une fonctionnalité non développée entrainera des pénalités financières. Ainsi, nous pouvons nous demander si pour garantir une adoption plus large de l'agilité en entreprise, il ne faudrait pas mener une réflexion approfondie sur les aspects contractuels. Pour nuancer, nous avons pu constater que l'utilisation d'une partie des pratiques et outils d'une méthode agile permettent déjà d'améliorer le fonctionnement d'une entreprise.

Par ailleurs, nous avons dressé plusieurs préconisations permettant à une entreprise qui souhaite se lancer dans une approche agile de le faire de façon progressive en commençant notamment par l'utilisation des outils avant les pratiques. Nous avons aussi mis l'accent sur la nécessité de bien évaluer la pertinence de la démarche en fonction du bénéfice que l'entreprise peut en tirer. Nous avons proposé pour cela l'utilisation de la méthode Méhari afin d'évaluer le niveau de criticité des dysfonctionnements de l'entreprise dans sa gestion de projet et de voir si l'agilité est une réponse adaptée.

D'autre part, nous avons développé l'application ASIPAT pour nous permettre d'améliorer la qualité de nos produits et la communication avec nos clients. Cette dernière apporte déjà une réponse aux pratiques d'ingénierie logicielle concernant les tests fonctionnels ; il reste à développer les fonctionnalités liées au suivi des projets agiles, chantier sur lequel nous allons nous atteler dans les prochains mois.

D'un point de vue pragmatique, je considère ce projet comme un succès car j'ai pu répondre aux quatre principaux besoins de l'entreprise que nous avons listés en page précédente. D'une manière plus large, le travail effectué dans le cadre de ce mémoire a été l'occasion de mettre en pratique les acquis de mes enseignements au CNAM, mais aussi de faire des rencontres enrichissantes avec des personnes expertes dans leur domaine pour m'aider à élaborer des solutions techniques et organisationnelles aux problèmes rencontrés.

Annexes

1. Le manifeste agile

1.1. Les quatre valeurs du manifeste

Valeurs extraites du site web officiel du manifeste agile :

http://agilemanifesto.org/iso/fr/

Les individus et leurs interactions plus que les processus et les outils ;

Des logiciels opérationnels plus qu'une documentation exhaustive ;

La collaboration avec les clients plus que la négociation contractuelle ;

L'adaptation au changement plus que le suivi d'un plan.

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.

1.2. Les principes sous-jacents au manifeste Agile

Principes tirés du site web officiel du manifeste agile :

http://agilemanifesto.org/iso/fr/principles.html

- *Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.*
- *Accueillez positivement les changements de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un avantage compétitif au client.*
- *Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.*
- *Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.*
- *Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.*
- *La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.*
- *Un logiciel opérationnel est la principale mesure d'avancement.*
- *Les processus agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.*
- *Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.*
- *La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.*
- *Les meilleures architectures, spécifications et conceptions émergent d'équipes auto organisées.*
- *À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.*

2. Réponses au questionnaire mené auprès des clients de CosiWeb

Réponses au questionnaire mené en novembre 2014 auprès des clients de CosiWeb, sur leur prédisposition à l'agilité.

Nombre de répondants	**22**

Client du public	68%
Client du privé	32%

Triangle de la performance[21]			
Classement	**1**	**2**	**3**
Conformité du site web aux attentes	55%	32%	14%
Respect du budget	18%	45%	36%
Respect des délais	9%	18%	73%

Au moment de signer le contrat	**Oui**	**Non**	**Je ne sais pas**
Accepteriez-vous de contractualiser avec un budget et une date d'échéance ferme, mais ne pas avoir forcément toutes les fonctionnalités souhaitées à échéance du projet?	23%	50%	27%
Accepteriez-vous de contractualiser pour un site web qui possèdera toutes les fonctionnalités souhaitées, mais ne pas avoir d'échéance ferme avec un budget qui pourra être révisé à intervalle régulier ?	18%	77%	5%
Préférez-vous avoir un budget ferme, une date d'échéance ferme et toutes les fonctionnalités souhaitées au départ, mais sans possibilité de les modifier durant toute la conception du site web ?	27%	59%	14%

Durant le développement de votre site web	**Oui**	**Non**	**Je ne sais pas**
Aimeriez-vous pouvoir ajouter de nouvelles fonctionnalités non prévues initialement ?	100%	0%	0%
Aimeriez-vous avoir la possibilité de modifier les fonctionnalités initiales, voire de les supprimer si elles ne correspondent plus à votre besoin ?	95%	5%	0%
Aimeriez-vous être impliqué dans sa conception en travaillant avec l'équipe à intervalles réguliers : validation des fonctionnalités, tests du site web ?	77%	14%	9%

En sachant que les fonctionnalités les plus importantes seront développées en priorité.	**Oui**	**Non**	**Je ne sais pas**
Accepteriez-vous que le site web ne possède pas toutes les fonctionnalités demandées mais juste celles qui sont essentielles ?	23%	59%	18%
Aimeriez-vous pouvoir arrêter le projet en cours de route lorsque vous jugez que les fonctionnalités essentielles sont développées ?	45%	23%	32%

[21] La somme des pourcentages peut dépasser 100 car les clients avaient la possibilité de classer les réponses en ex aequo.

Bibliographie

Aubry, C. (2013). *Scrum - 3e éd. - Le guide pratique de la méthode agile la plus populaire* (Édition : 3e édition). Paris: Dunod.

Audience de l'Internet mobile. (2014). Retrieved from http://www.mediametrie.fr/internet/solutions/l-audience-de-l-internet-mobile.php?id=93

Ayari, M. (2013). Cours IPST CNAM RSX112: Sécurité et réseaux.

Batatia, H. (2013). Cours IPST CNAM ENG221 partie B: Information et communication pour l'ingénieur.

Beaugrand, T., & Belin, J.-B. (2013). Le contrat de développement logiciel en méthode Agile. *Expertises*, 415–420.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change, 2nd Edition* (2nd edition). Boston, MA: Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifeste pour le développement agile de logiciels. Retrieved from http://agilemanifesto.org/iso/fr/

Boisvert, M., & Trudel, S. (2011). *Choisir l'agilité - Du développement logiciel à la gouvernance*. Paris: Dunod.

Brotschi, J.-J. (2011). Cours IPST CNAM EME102: Management de l'entreprise.

Cohn, M. (2009). *Succeeding with Agile: Software Development Using Scrum* (1 edition). Upper Saddle River, NJ: Addison-Wesley Professional.

Deuff, D., & Cosquer, M. (2013). *Méthode agile centrée utilisateurs*. Cachan: Lavoisier.

Fernandez, V., Houy, T., & Khalil, C. (2013). *Les méthodes agiles de développement informatique*. Paris: Presses des MINES.

Highsmith, J. (2001). History: The Agile Manifesto. Retrieved from http://agilemanifesto.org/history.html

Hohmann, L. (2006). *Innovation Games: Creating Breakthrough Products Through Collaborative Play* (Édition : 1). Upper Saddle River, NJ: Addison Wesley.

Legardeur, L., & Sutherland, J. (2009). Enquête Nationale Scrum User Group. Retrieved from http://www.frenchsug.org

Legeard, B. (2014). Test fonctionnel et Développement Agile. Retrieved from http://www.cftl.fr/?id=71

Messager, V., & Tabaka, J. (2010). *Gestion de projet agile, avec Scrum, Lean, Extreme Programming...* Paris: Eyrolles.

Meyer, B. (2013). *Agile ! Le Bon, le Bête et le Méchant*. Palaiseau - Ecole Polytechnique. Retrieved from http://webcast.in2p3.fr/videos-bertrand_meyer

Poiron, Y. (2010). Comprendre SOAP – Partie 1. Retrieved from http://www.blog-nouvelles-technologies.fr/827/comprendre-soap-partie-1/

Royce, W. W. (1970). Managing The Development of Large Software Systems (p. 330). Presented at the Proceedings of IEEE WESCON 26. Retrieved from http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

Schwaber, K. (1995). Scrum Development Process. Retrieved from http://www.jeffsutherland.org/oopsla/schwaber.html

Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum* (1 edition). Upper Saddle River, NJ: Prentice Hall.

Schwaber, K., & Sutherland, J. (2012). *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust*. John Wiley & Sons.

Scrum Overview for Agile Software Development. (2005). Retrieved from http://www.mountaingoatsoftware.com/agile/scrum/overview

Stackelberg, H. von. (1934). *Marktform und Gleichgewicht, von Heinrich von Stackelberg*. Wien: J. Springer.

Thomas, D. (2013). Tutoriel sur le test d'applications Web avec Selenium. Retrieved from http://atatorus.developpez.com/tutoriels/java/test-application-web-avec-selenium/

Valentine, C. E. (2011). *Comparison of System Development Methodologies*. Retrieved from http://www.academia.edu/7361851/Comparison_of_System_Development_Methodologies

Vernois, A. (2014). *Unit test : After, Before and TDD*. Agile Tour Toulouse.

Zorzabalbère, B. (2011). Cours IPST CNAM ENG110: Management de projet pour l'ingénieur.

Liste des figures

Liste des tableaux

Mise en œuvre de l'agilité au sein de la société CosiWeb

Mémoire d'Ingénieur C.N.A.M. en informatique

Jean-Pierre FONTES **Soutenu à Toulouse, le 22 mai 2015**

Résumé

Afin de mener à bien un développement informatique, les méthodes agiles proposent un modèle qui consiste à découper le projet en plusieurs étapes d'une durée de quelques semaines, nommées « *itérations* » et de livrer à la fin de chacune d'entre elles une version fonctionnelle du produit, ce dernier étant ainsi conçu de façon « *incrémentale* ». En outre, l'approche agile part du constat que les exigences du client peuvent évoluer durant le cycle de vie du projet et qu'il est nécessaire de prendre en compte ces changements afin de garantir son succès. Le premier objectif de ce mémoire est d'évaluer les besoins de l'entreprise CosiWeb qui peuvent être comblés par des pratiques agiles. Le second objectif consiste à expérimenter l'agilité autour de la gestion de projet puis de concevoir une application web de suivi et de gestion de tests fonctionnels afin d'améliorer la qualité logicielle des produits conçus par l'entreprise.

Mots clefs : méthode agile, test fonctionnel, SOAP, méhari, innovation games.

Summary

This report deals with the agile methods in the context of data-processing developments. The agile methods offers a model which consists in cutting the project in several few week stages (named "iterations") and delivering at the end of each iteration a functional version of the product (the latter being designed in an incremental way). Besides, the agile approach relies on the fact that the customers' requirements can evolve during the life cycle of the project and that these changes need to be taken into account to guarantee the success of the very project. The first goal of this report is to estimate CosiWeb's needs and how agile practices can meet them. The second one consists in experimenting agile methods on project management. The last one is to design a web application of functional test follow-up and management to improve the software quality of the products designed by the company.

Keywords : agile software development, functional testing, SOAP, méhari, innovation games.