TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AHP	Analytic Hierarchy Process
API	Application Programming Interface
ASD	Adaptive software development
DSDM	Dynamic Systems Development Method
ESCAM	Extended Scrum Cloud Agile Method
FDD	Feature Driven Development
GSD	Global Software Development
IaaS	Infrastructure as a Service
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
PMBOK	Project Management Body of Knowledge
SaaS	Software as a Service
SBOK	Scrum Body of Knowledge
TaaS	Testing as a service
UP	Unified Process
XP	eXtreme Programming

INTRODUCTION

Le cloud computing «L'informatique en nuage», est aujourd'hui une technologie qui est vue à cause de ces nombreux avantages. Cette technologie permet d'offrir à des coûts assez bas aux utilisateurs (entreprises, clients, développeurs, particuliers) des ressources informatiques mutualisées et accessibles de n'importe où. Il permet donc aux entreprises de développement logiciel d'offrir plus facilement et à moindre coût aux clients et utilisateurs des applications à tout instant à travers les navigateurs web d'autres applications dédiées. Suivant le service du SaaS (Software as Service) les données et les applications sont localisées dans le nuage informatique. Ceci facilite la mise à l'échelle rapide des applications. Ces dernières sont pour la plupart développées suivant les méthodes de développement logiciel actuelles (méthodes agiles : XP (eXtreme Programming), Scrum, UP (Unified Process), DSDM (Dynamic Systems Development Method), FDD (Feature Driven Development), ASD (Adaptive software development) et celles traditionnelles ou hybrides. Après le développement, ces applications sont déployées dans le cloud. L'utilisation des services du cloud change donc le cycle de vie du développement des applications, car les méthodes agiles habituelles ne prennent pas en compte toutes ces contraintes et avantages. Les méthodes de développement logiciel constituent un des facteurs importants dans le succès des projets. En occurrence, l'utilisation de processus agile et l'implication des utilisateurs (pratique de l'agilité) sont deux facteurs majeurs dans le succès des projets TI selon rapport Chaos Manifesto (The Standish Group International, 2013).

À cet effet, il est important d'étudier le processus de développement des applications dans le cloud afin d'adapter les méthodes agiles usuelles pour assurer le succès des projets de développement et accroître la satisfaction des clients et utilisateurs (un aspect essentiel dans le principe de l'agilité). Également, l'utilisation des services de l'informatique est considérée comme une solution pour certains problèmes comme les pertes de temps dans le déploiement des livrables à chaque itération (Abhishek, Reena, & Rani, 2012), (Kalem, Donko, & Boskovic, 2013a).

C’est à cet égard que ce mémoire est consacré à l’applicabilité des méthodes agiles de développement logiciel dans le cloud. La méthode Scrum qui est l’une des plus utilisées actuellement est adaptée aux contextes du développement d’application dans la couche PaaS (Plat-form as a Service) du cloud.

Mise en contexte et justification

L’avènement de l’informatique en nuage a entraîné d’énormes changements dans l’exploitation des logiciels. Les applications sont de plus en plus développées pour être hébergées dans le cloud. Les fournisseurs de services proposent aux clients, des applications directement utilisables dans le cloud. Les utilisateurs, clients et même les entreprises n’ont plus besoin de télécharger et d’installer ces applications en local. Les applications du cloud les plus populaires sont Gmail, Google Docs, Microsoft 365, etc. Pour mettre ses services à disposition des utilisateurs, les fournisseurs développent ces applications et les déploient dans le cloud. Ces fournisseurs de services proposent de plus en plus aux entreprises et aux développeurs des API (Application Programming Interface), services et plates-formes le développement des applications dans le cloud (par exemple Google Engine, IBM Bluemix, Microsoft Azure, Amazon AWS). Ceci implique donc des changements dans le cycle de vie des projets de développement logiciel. Les méthodes de développement logiciel actuelles doivent être donc adaptées aux contextes de l’utilisation des services du cloud. Les méthodes agiles étant les plus utilisées aujourd’hui pour le développement logiciel, il est donc important de les étudier afin de savoir celles qui répondent le plus aux exigences du développement dans le cloud. Aussi, il est question de sélectionner une méthode agile et l’adapter au développement d’applications dans le cloud.

Pour résoudre ce problème, certains auteurs ont travaillé sur ce sujet. En occurrence les travaux de (Patidar, Rane, & Jain, 2011) qui abordent les challenges du développement logiciel dans le cloud. Ces challenges concernent : l’hétérogénéité des plates-formes et services web ; les entreprises ayant des équipes de développement dispersées partout dans le monde; l’inadéquation des modèles de processus de développement logiciels existants aux contraintes

des fournisseurs de services du cloud. À cet effet, ils proposent l'intégration du rôle «cloud provider» aux différentes phases du cycle de vie du développement logiciel afin de répondre aux questions suivantes : 1) combien de développeurs a-t-on besoin, 2) les composants réutilisables, 3) l'estimation des coûts, 4) l'estimation des délais, 5) la gestion des risques, 6) la gestion des configurations, 7) la gestion des changements, 8) l'assurance qualité.

Le développement des applications par des équipées situées sur des sites géographiquement éloignés (GSD – Global Software Development) est une problématique majeure que d'autres auteurs ont abordée. En effet, selon (Hashmi et al., 2011), le développement d'applications offshores fait face à quatre challenges majeurs : 1) géographique, 2) culturel, 3) linguistique et 4) décalage temporel entre les membres d'équipes, 5) niveau d'expérience. Ces auteurs recommandent l'utilisation de la plate-forme de cloud (PaaS) pour le développement des applications. (Cocco, Mannaro, & Concas, 2012) a proposé une méthode Srcum pour l'optimisation du processus de développement d'applications offshores en utilisant la méthode de Système Dynamique de Jaw W. Forrester. (Chung & Shao-Zong, 2013) ont travaillé sur la conception d'un système de gestion du développement des logiciels distribués dans le cloud appelé DDMan. Ce modèle est basé sur celui du WebSD en utilisant le service du PaaS. DDMan formalise les communications et définit les processus et les rôles des différents membres d'équipes (programmeurs, testeurs, managers, etc.).

Avec l'évolution des services offerts par les fournisseurs de services, les développeurs et gestionnaires peuvent se passer de la présence du représentant du fournisseur (comme recommandé par (Patidar et al., 2011)) pour la conduite et la réalisation des projets en se basant sur les guides de référence **ISO/IEC 17788:2014** et **ISO/IEC 17789:2014**. Ces guides définissent les différents acteurs, rôles et activités dans le cloud, ainsi que ses composantes.

Objectifs et résultats attendus

Le but principal de cette recherche est de vérifier l'applicabilité du développement agile de logiciel dans le cloud. Ce qui conduit aux objectifs suivants :

- Démontrer que l'utilisation des méthodes agiles de développement dans le cloud offre de multiples avantages ;
- Déterminer et analyser les couches (SaaS, PaaS et IaaS) concernées par le développement ;
- Concevoir une méthode Scrum adaptée aux contextes de développement d'applications dans l'environnement du PaaS.

Ce travail a une importance majeure non seulement pour les entreprises de développement logiciel, mais aussi pour les utilisateurs et les fournisseurs de services du cloud. Il permettra aux entreprises de développement d'avoir plus d'informations pour faciliter la migration de leurs activités de développement dans le cloud. Ces entreprises disposeront donc d'un ensemble de modifications à apporter à la méthode de développement Scrum dans le cloud et pourront facilement évaluer et sélectionner les services. En dernier lieu, tout ceci leur permettra aussi de réduire les coûts de développement tout en livrant des logiciels de qualité supérieure.

Méthodologie de recherche

1. **Partie 1** : analyse du développement d'applications dans le cloud suivant le guide de référence **ISO/IEC 17789:2014;**
2. **Partie 2** : élaboration de la méthode de développement Scrum dans le cloud;
3. **Partie 3** : description des composantes de la méthode Scrum adaptée au développement d'applications dans le PaaS et comparaison avec d'autres méthodes ou développement dans le cloud.

Organisation du mémoire

Ce mémoire est organisé en 8 chapitres. Il commence par la présentation des concepts de base sur le cloud computing et le développement logiciel agile. Le chapitre 2 aborde les travaux de recherches effectués sur le développement de logiciels dans le cloud. Dans ce chapitre, les solutions, modèles et guides proposés par certains auteurs ont été présentés et analysés. Le chapitre 3 aborde la description de l'approche DevOps ainsi que son adoption dans le cloud. Quant à lui le chapitre 4 fait l'analyse du processus développement d'applications en proposant les caractéristiques fondamentales d'un modèle de processus de développement dans le cloud. Aussi, les différentes couches du cloud ont été analysées et l'environnement du PaaS a été sélectionné dans le cadre de ce mémoire. Dans le chapitre 5, la méthode Scrum est sélectionnée et adaptée à l'environnement du PaaS. Cette méthode est décrite à travers un ensemble d'évènements, de rôles et pratiques. Chaque évènement ou activité est présenté tout en précisant l'implication des services du PaaS dans la gestion des projets agiles. le chapitre 6 est consacré à l'analyse de l'impact de l'utilisation du PaaS sur les outils de gestion de Scrum. Vers la fin, les derniers chapitres ont fait la synthèse des contributions de cette recherche, ses limites et recommandations pour de futurs travaux.

CHAPITRE 1

CONCEPTS DE BASE

1.1 Introduction

La réussite de ce sujet de recherche passe par la compréhension de ces concepts fondamentaux. Il est donc question de présenter dans un premier temps l'informatique en nuage, ses caractéristiques et composantes. Par la suite, il s'agit d'aborder l'agilité et ses méthodes usuelles dans le domaine du développement logiciel.

Ce chapitre montre les différents aspects liés à l'utilisation du cloud et au développement logiciel par les méthodes agiles.

1.2 Le cloud et ses caractéristiques

1.2.1 Définition

L'informatique en nuage est un modèle permettant un accès pratique et sur demande à un ensemble de ressources partagées et configurables rapidement(ISO/IEC, 2014a, 2014b). Ces ressources sont gérées avec un minimum d'effort. Selon le NIST (National Institute of Standards and Technology) et la norme ISO/IEC 17788 :2014 l'informatique en nuage possède six caractéristiques essentielles, trois types d'offres de services et quatre modèles de déploiement (ISO/IEC, 2014a, 2014b) et (Peter & Timothy, 2011). Le guide de référence ISO/IEC 17789 définit les différents éléments, de l'architecture du cloud. Il offre aux utilisateurs un ensemble à moindre coût et extensible rapidement avec le moins d'effort que possible.

1.2.2 Caractéristiques

Le cloud computing est une technologie caractérisée essentiellement par :

- **Service à la demande** : le client ou l'utilisateur peut gérer et consommer les ressources informatiques sans l'intervention humaine du fournisseur de services ;
- **Accès au réseau :** permettre la disponibilité et l'accès au réseau en utilisant des plates-formes clientes hétérogènes tels les téléphones mobiles, tablettes, ordinateurs portables, postes de travail ;
- **Ensemble de ressources :** les ressources sont regroupées pour desservir plusieurs consommateurs avec différentes ressources physiques et virtuelles. Ces ressources sont attribuées dynamiquement en fonction du besoin du client. Exemples de ressources de stockage, le traitement, la mémoire, et la bande passante du réseau ;
- **Élasticité rapide** : c'est la capacité qu'a une application de passer à l'échelle de manière dynamique et rapide en fonction des besoins. En ce qui concerne le consommateur, les ressources utilisées semblent être illimitées et peuvent être louées à n'importe quel moment et en quantité illimitée ;
- **Service sur mesure** : contrôle automatique et optimisation des ressources allouées. Ce mécanisme s'opère à un certain niveau d'abstraction approprié pour le type de service (par exemple, stockage, traitement, bande passante) utilisé ;
- **Multi-location** : permettre le partage de ressources mutualisées par plusieurs clients. Chaque client instancie virtuellement les ressources et les paramètres en fonction de ses besoins.

1.2.3 Catégories de services

L'architecture des services du cloud est subdivisée en trois couches essentielles (Application, Plate-forme et Infrastructure). Trois grandes catégories de services de base de l'informatique en nuage sont énumérées suivant la norme ISO/IEC 17788:2014 :

- **Infrastructure en tant que Service (Infrastructure as a Service - IaaS)** : C'est la couche de base de l'informatique en nuage. Les ressources telles que (CPU, stockage, mémoire, réseau) sont fournies à la demande du consommateur pour le déploiement et l'exécution de ses applications (incluant les systèmes d'exploitation) (Etchevers, 2012), (Peter & Timothy, 2011). Amazon EC2, Windows Azure, Google Compute Engine et Rackspace sont des exemples de fournisseurs de service IaaS.

- **Plate-forme en tant que Service (Plat-form as a Service - PaaS)** : Elle est au-dessus de la couche de service IaaS. Elle fournit aux développeurs des outils tels que les langages de programmation, les API, les bibliothèques nécessaires pour la création et le déploiement des applications dans l'informatique en nuage. Le consommateur ne gère ni ne contrôle l'infrastructure de nuage sous-jacente, y compris le réseau, les serveurs, les systèmes d'exploitation, ou l'espace de stockage, mais il a le contrôle sur les applications déployées et les paramètres de configuration pour l'environnement d'application d'hébergement (ISO/IEC, 2014a). Il existe plusieurs acteurs tels que Salesforce.com, Google App Engine, CloudBees, Heroku et CloudFoundry qui fournissent des services de la couche PaaS.

- **Logiciel en tant que service (Software as service – SaaS)** : C'est la dernière couche de service proposé par l'informatique en nuage. Elle fournit aux consommateurs des applications accessibles à partir de divers périphériques clients par le biais soit d'une interface client légère, comme un navigateur Web (par exemple, webmail), ou d'une interface de programme. Le consommateur ne gère ni ne contrôle l'infrastructure sous-jacente, y compris nuage réseau, serveurs, systèmes d'exploitation, de stockage, ou même les capacités individuelles d'application, à l'exception possible des paramètres de configuration d'applications limitées (ISO/IEC, 2014a). Gmail, Google Docs, DrpoBox sont des exemples d'application opérant au niveau de la couche SaaS.

D'autres catégories de services ont été créées afin de répondre aux besoins des entreprises et des utilisateurs. Le tableau ci-dessous présente quelques variantes de services.

Tableau 1.1 Liste des services en fonction des couches du cloud
Tirée de (ISO/IEC, 2014a)

Cloud service categories	**Cloud capabilities types**		
	Infrastructure	**Platform**	**Application**
Compute as a Service	X		
Communication as a Service		X	X
Data Storage as a Service	X	X	X
Infrastructure as a Service	X		
Network as a Service	X	X	X
Platform as a Service		X	
Software as a Service			X

1.2.4 Modèles de déploiement

Quatre modèles de déploiement sont utilisés :

- **Nuage privé (Private cloud)** *:* Le nuage privé est exploité uniquement par une organisation. La plupart du temps, il est hébergé en son sein. Il peut être géré par l'organisation elle-même ou par un tiers. En d'autres mots, un nuage privé est une architecture informatique propriétaire ou dédiée à un nombre limité d'utilisateurs, derrière un pare-feu pour assurer le contrôle et la gestion (ISO/IEC, 2014a), (Paraiso, 2014).

- **Nuage communautaire (Community cloud)** : Le nuage communautaire est un modèle qui est partagé entre plusieurs organismes afin de répondre à des besoins spécifiques (par exemple, mission collaborative, sécurité, politique). Le nuage communautaire est conçu pour répondre aux exigences d'une communauté ou d'autres types d'entreprises particulières (prestataire pour l'armée, recherche, etc.). À titre d'exemple, les entreprises qui travaillent dans des domaines sensibles comme l'armement et qui ont un réseau

similaire peuvent avoir besoin de se réunir sur un nuage communautaire (ISO/IEC, 2014a), (Paraiso, 2014).

- **Nuage public (Public cloud) :** Le nuage public est mis à la disposition du grand public qui peut être un grand groupe ou une PME. Il est géré par un organisme appelé fournisseur de nuage qui vend des services de nuage. C'est le modèle avec lequel les services de nuage sont fournis et tout le monde est libre d'y souscrire. Par exemple, Amazon EC2, Windows Azure et Rackspace sont des fournisseurs d'infrastructure de nuages publics (Paraiso, 2014).
- **Nuage hybride (Hybrid cloud) :** Le nuage hybride, comme son nom l'indique, est la combinaison de deux ou plusieurs types de nuages. Une organisation peut créer un nuage privé pour ses applications qui sont étroitement intégrées à des systèmes patrimoniaux existants et avoir des exigences de mise à l'échelle que ne peut satisfaire le nuage privé. Ainsi, l'organisme peut exploiter le nuage public en créant un lien entre les deux nuages (ISO/IEC, 2014a), (Paraiso, 2014).

1.2.5 Rôles et activités

La norme ISO/IEC 17789:2014 présente les parties prenantes, les activités, rôles et les composantes fonctionnelles. Elle définit également les interactions entre ces éléments dans l'environnement de l'informatique en nuage.

Rôles

Un rôle constitue un ensemble d'activités de l'environnement de l'informatique en nuage qui poursuivent un objectif commun. Trois rôles sont définis par la norme ISO/IEC 17789:2014 :

- **Cloud service customer (CSC)** : une partie prenante de la relation d'affaires qui utilise les services de l'informatique ne nuage ;

- **Cloud service provider (CSP)**: cette catégorie de partie prenante rend disponible les services de l'informatique en nuage;
- **Cloud service partner (CSN)**: concerne la partie engagée dans le soutien des activités des fournisseurs et utilisateurs des services de l'informatique ou les deux.

La norme ISO/IEC 17789:2014 définit également les sous-rôles suivant le schéma ci-dessous :

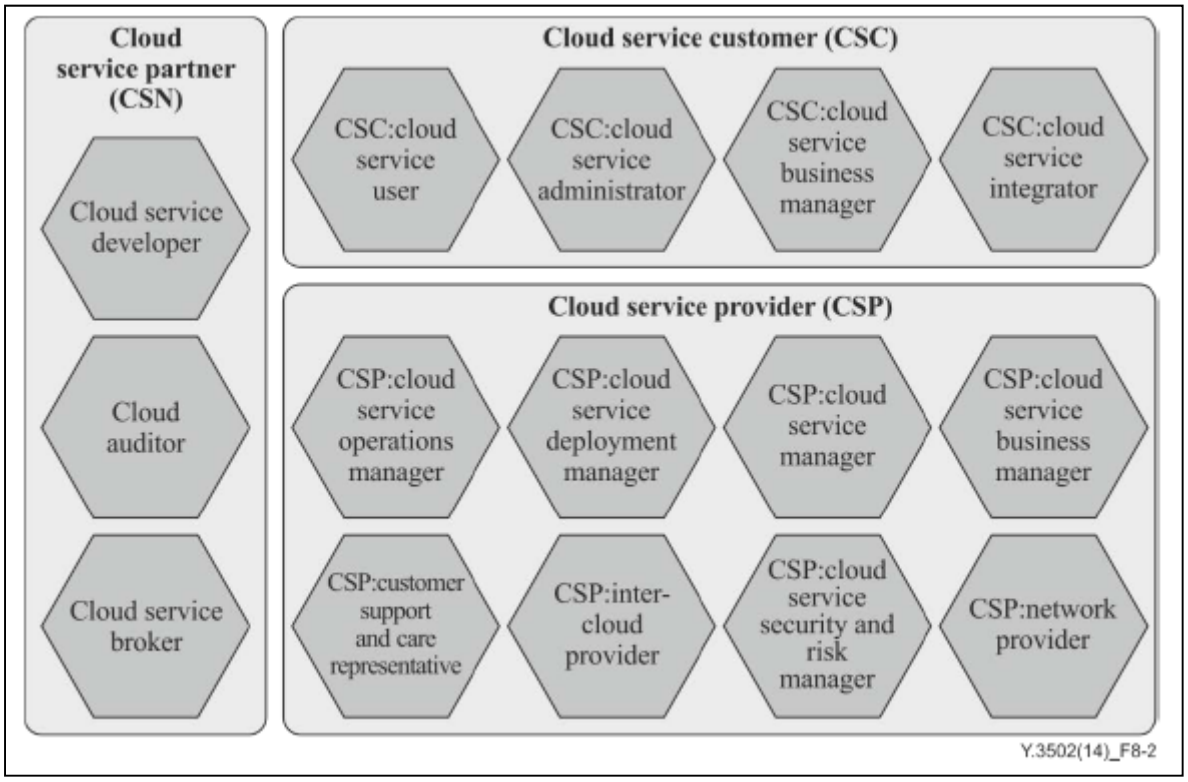

Figure 1.1 Les rôles et sous-rôles du cloud
Tirée de (ISO/IEC, 2014a)

Le « **cloud service developer** » est un sous rôle du « **cloud service partner** » (CSN) qui s'occupe de la conception, du développement, des tests et de la maintenance des composantes ou applications (ISO/IEC, 2014b). Il intervient essentiellement au niveau de la couche PaaS.

Activités

Une activité est un ensemble de tâches utilisant des composantes fonctionnelles pour livrer un ou plusieurs résultats. Chaque rôle ou sous-rôle définit un ensemble d'activités. La figure ci-dessous présente les activités des rôles du « ***cloud service partner*** **» (CSN).**

Relation entre les rôles

Les interactions entre les rôles permettent de mettre en exergue liens entre ces derniers et les composantes fonctionnelles nécessaires. La norme ISO 17789:2014 présente les relations entre les différents rôles du cloud. Le schéma ci-dessous présente la relation fonctionnelle entre le « ***cloud service developer*** **»** et le « ***cloud service provider*** **»**. Cette interrelation permet de visualiser les composantes fonctionnelles nécessaires pour les activités de développement d'applications dans le cloud.

1.3 Les méthodes de développement agiles

1.3.1 L'agilité

En gestion de projet, la méthode en Cascade fut l'une des premières méthodes utilisées en raison de sa simplicité et de son caractère séquentiel et linéaire. Cette méthode convient plus aux projets dont les besoins et exigences sont bien définis au début du projet. Les demandes de changements ne sont pas souvent les bienvenues.

Les projets de développement logiciel sont souvent caractérisés par des besoins non totalement définis et des demandes de changements qui surviennent à tout moment dans le projet. Cette complexité a conduit progressivement à l'adoption des méthodes agiles dans le domaine de l'ingénierie logicielle. Les méthodes agiles visent fondamentalement la grande satisfaction des parties prenantes du projet (les clients, les utilisateurs, les membres d'équipe, la direction, etc.).

Suivant le «manifeste agile», on retient la définition suivante : « **Une méthode agile** est une approche itérative et incrémentale pour le développement de logiciel, réalisée de manière très collaborative par des équipes responsabilisées, appliquant un cérémonial minimal, qui produisent, dans un délai contraint, un logiciel de grande qualité répondant aux besoins changeants des utilisateurs ». (Tekool.net, 2016).

Selon le manifeste agile, une méthode agile possède quatre valeurs et 12 principes :

Valeurs

- **Les individus et les échanges** plus que processus et des outils ;
- **Le produit** plus que de la documentation excessive ;
- **La collaboration du client** plus que la négociation ;
- **La réactivité** plus que le suivi d'un plan

Principes

- Satisfaire le client en livrant tôt et régulièrement des logiciels utiles, qui offrent une véritable valeur ajoutée ;
- Accepter les changements, même tard dans le développement ;
- Livrer fréquemment une application qui fonctionne ;
- Collaborer quotidiennement entre clients et développeurs ;
- Bâtir le projet autour de personnes motivées en leur fournissant environnement et support, et en leur faisant confiance ;
- Communiquer par des conversations en face à face ;
- Mesurer la progression avec le logiciel qui fonctionne ;
- Garder un rythme de travail durable ;
- Rechercher l'excellence technique et la qualité de la conception ;
- Laisser l'équipe s'auto organiser ;

- Rechercher la simplicité ;
- À intervalles réguliers, réfléchir aux moyens de devenir plus efficace.

1.3.2 Les méthodes agiles en développement logiciel

Cette partie présente les méthodes agiles de développement logiciel les plus usuelles et les plus répandues.

1.3.2.1 eXtreme Programming (XP)

XP se définit comme un ensemble de pratiques qui vise à se consacrer à la réalisation elle-même. Ces pratiques sont essentiellement axées la programmation permettant d'améliorer continuellement la conception et code. Voici quelques pratiques d'ingénierie logicielle comme :

- **Le développement piloté par les tests** : les tests unitaires sont développés avant le développement des fonctionnalités ;
- **La programmation en binôme** : XP encourage le développement à 2 afin de faciliter la détection les anomalies et erreurs éventuelles.

1.3.2.2 Dynamic Systems Development Method (DSDM)

La méthode DSDM a été créée en 1994 pour répondre aux besoins de qualité et aux principes de RAD (Rapid Development Application). Elle permet donc la livraison rapide et efficace des résultats (DSDM Consortium, 2014). La méthode AgilePF DSDM est conçue pour permettre l'intégration facile des autres approches agiles comme Scrum et XP. Elle est utilisée pour faciliter l'accessibilité des projets par les parties prenantes. Les phases de développement de la méthode AgilePF sont :

- ***Pre-Project* :** pour s'assurer que le projet est aligné avec les stratégies de l'entreprise et répond aux objectifs d'affaires ;

- ***Feasability* :** pour vérifier si le projet est réalisable techniquement et s'il est rentable du point de vue commercial ;
- ***Foundations* :** elle permet de s'assurer de la compréhension de la portée du projet et détermine le cycle de vie du projet ainsi que les processus du modèle à utiliser. Elle peut durer plusieurs semaines pour les gros projets ;
- ***Evolutionary Development* :** utilisation d'une approche de développement incrémentale et itérative ;
- ***Deployment* :** phase finale d'assemblage des différents livrables. Livraison et déploiement du produit ;
- ***Post-project*** : pour s'assurer que les objectifs commerciaux ont été atteints.

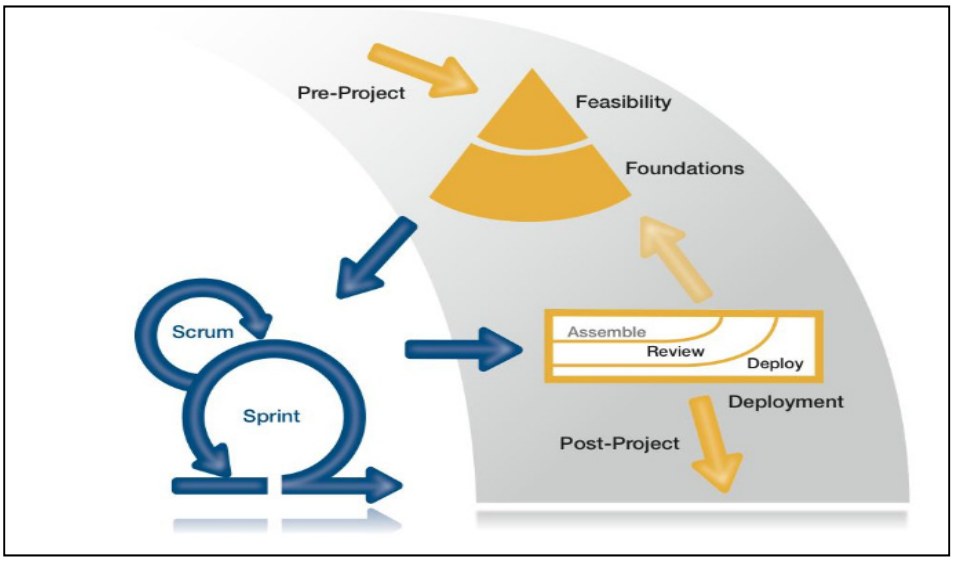

Figure 1.2 Phases de développement de AgilePF
Tirée de (DSDM Consortium, 2014)

Chaque projet a un cycle de vie qui commence par l'identification d'un besoin potentiel et se termine à un moment où ce besoin est soit terminé ou a été rejeté (DSDM Consortium, 2014). Le rejet peut se produire à tout moment si le projet devient non viable d'un point de vue commercial. La phase post-project permet à AgilePF de tenir compte de l'après-projet, ce que

ne fait pas habituellement la méthode Scrum. AgilePF tient donc compte de 2 éléments que sont **la gestion du projet** et **la livraison du produit.**

La méthode DSDM est plus formelle avec un nombre assez important de rôles, de processus et d'artéfacts. Elle est aussi caractérisée par la méthode de classification et de priorisation des exigences suivant la méthode **MoSCoW** (DSDM Consortium, 2014; Hibbs, Jewett, & William, 2010) :

- **M** (Must have) : fonctionnalités obligatoires ;
- **S** (Should have) : fonctionnalités importantes à faires si possibles ;
- **C** (Could have) **:** fonctionnalités possibles, mais non indispensables ;
- **W** (Won't have) : fonctionnalités qui peuvent attendre la prochaine fois.

1.3.2.3 Scrum

La méthode Scrum est la méthode agile la plus populaire. Elle est définie par son fondateur Ken Schwaber comme un cadre (framework) qui présente les éléments qui feront partie du processus appliqué pour la réalisation d'un produit. Scrum impose des itérations de courtes durées appelées ***Sprint*** pour le développement du produit. Le schéma ci-dessous montre le déroulement d'un ***Sprint.***

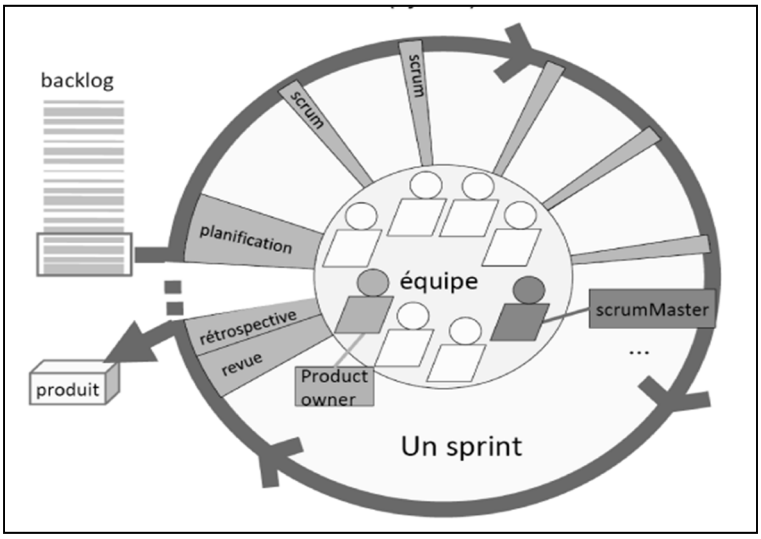

Figure 1.3 Schéma de déroulement d'un Sprint

Tirée de (Aubry, 2014)

Les caractéristiques

- Les fonctionnalités à réaliser sont définies et regroupées par ordre de priorité dans le ***Backlog*** de produit par le ***Product Owner***;
- À chaque itération ou ***Sprint*** de durée courte et fixe et définie à l'avance, des fonctionnalités sont développées pour créer une version du produit appelée ***Release.*** Le contenu de chaque itération est défini par le ***Product Owner***;
- Au cours des ***Sprints*** des rencontres quotidiennes appelées **mêlées** sont organisées par le ***Scrum Master*** pour l'application des principes de Scrum et suivre l'avancement par rapport aux engagements afin d'assurer le succès du ***Sprint*** ;
- À la fin de chaque ***Sprint,*** un produit partiel est livré. Son évaluation et les rétroactions permettent d'ajuster le ***Backlog*** pour le prochain ***Sprint.*** À cette étape, l'équipe de projet identifie les anomalies afin d'améliorer le processus lors des prochains ***Sprints.*** Cette étape est très importante pour le succès du projet. Elle implique l'intervention du ***Product Owner***, du client ou des utilisateurs finaux.

Les composantes

Les composantes nécessaires à l'utilisation de la méthode Srcum sont présentées ci-dessous :

- **Rôles :** Définissent les acteurs clés de la méthode Scrum
- **Cérémonial** : Définissent les blocs de temps nécessaire pour créer la régularité dans les ***Sprints***
- **Artefacts :** Définissent les différents documents et outils. Le plus important est le ***Backlog***. Il faut noter que Scrum impose très peu d'artefacts.

Les phases de développement

En ingénierie logicielle, quatre phases sont habituellement utilisées pour le développement des produits :

- **S**pécification fonctionnelle (définition des exigences fonctionnelles);
- **A**rchitecture (conception);
- **C**odage (et test unitaire);
- **Te**st (d'intégration et de recette)

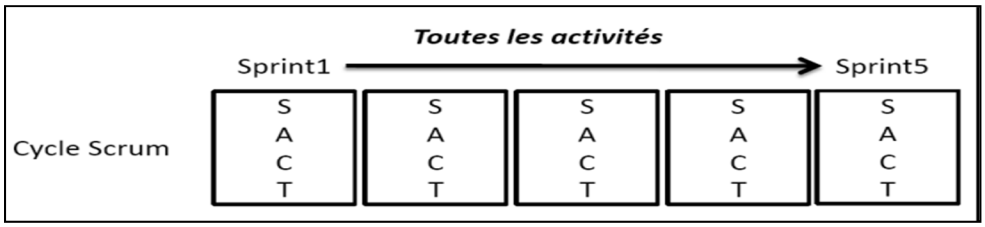

Figure 1.4 Cycle Scrum basé sur 5 Sprints – Itérations
Tirée de (Aubry, 2014)

Figure 1.5 Cycle séquentiel (Classique)
Tirée de (Aubry, 2014)

Les 2 figures ci-dessus montrent la différence fondamentale entre la méthode classique comme cascade et celle basée sur les principes de la méthode agile. Dans la méthode classique, les différentes activités sont séquentielles. Ce qui implique que les tests sont effectués à la fin du projet. En cas d'erreurs ou de problèmes liés à la conception ou à la définition des exigences, il est difficile de faire des changements. Les demandes de changement ne sont pas souvent acceptées. Ce qui entraîne des délais complémentaires pour la livraison du produit et l'insatisfaction du client dans certains projets.
Avec les méthodes agiles comme Scrum, toutes les activités formelles ou évènements sont exécutés durant chaque itération. Ce qui facilite la détection et la correction des erreurs et la prise en compte des demandes de changements.

1.3.3 Synthèse

Plusieurs méthodes agiles sont utilisées aujourd'hui pour les projets de développement d'applications. Ces méthodes se basent sur les valeurs et principes de la méthodologie agile pour livrer de façon incrémentale et itérative des applications fonctionnelles. Chaque méthode définit ses rôles, artéfacts et pratiques afin de faciliter le développement et la livraison des applications. La méthode Scrum est l'une des plus utilisées en raison de sa simplicité.

1.4 Conclusion du chapitre

Après avoir passé en revue les différents aspects et caractéristiques du nuage, plusieurs méthodes agiles de développement agiles ont été abordées. Il est à noter que la norme ISO/IEC 17789 :2014 définit les différents rôles et sous rôles nécessaires pour l'utilisation du cloud pour le développement logiciel. Ainsi, les « cloud service provider » mettent à disposition des « cloud service developer » l'environnement et les outils de développement pour leur faciliter la construction et le déploiement des applications.

À cet effet les modèles de services SaaS, IaaS et surtout PaaS sont les plus concernés par le développement d'applications dans le cloud. L'utilisation de ces modèles de services pour le développement engendre donc l'adaptation des méthodes agiles usuelles afin de répondre aux besoins des clients et utilisateurs. Le prochain chapitre aborde donc les travaux qui ont été effectués dans ce sens pour analyser et justifier l'applicabilité des méthodes agiles de développement dans le cloud.

CHAPITRE 2

REVUE DE LITTÉRATURE SUR LE DÉVELOPPEMENT LOGICIEL DANS LE CLOUD

2.1 Introduction

Avec l'importance du développement logiciel dans l'évolution des technologies aujourd'hui, plusieurs travaux de recherches ont abordé ce domaine. Dans le cadre de ce travail de recherche, la revue de littérature est faite sur l'agilité en gestion de projet dans un premier temps et sur le développement logiciel pour et dans le cloud. L'objectif de ce chapitre est d'identifier et de faire la synthèse des travaux de recherches en relation avec le sujet de recherche.

2.2 L'agilité en gestion de projet

2.2.1 Le succès des projets

En référence au PMBOK (Project Management Body of Knowledge) (Project Management Institute, 2013), le succès d'un projet est mesuré par la qualité du produit et du projet, le respect des délais, du budget, de la portée du niveau de satisfaction client.

La triple contrainte : Le respect des trois contraintes de base (Portée, Coût, et Délais) est une condition de base pour le succès d'un projet (Schwalbe, 2011) :

- ***Portée*** : définis les requis et le contenu du projet ;
- ***Coût*** : définis une estimation du coût de réalisation du projet ;
- ***Délai*** : définis une estimation du temps nécessaire à la réalisation du projet.

Afin d'atteindre les objectifs du projet, une ou plusieurs de ces contraintes sont ajustables en fonction de l'environnement. En mode agile, la portée est ajustable afin de livrer un produit de qualité répondant aux besoins des clients.

Le groupe « **The Standish group International** » définit le succès d'un projet comme l'atteinte des trois contraintes précédentes, à travers son rapport CHAOS 2013. Ainsi en 2012, 39 % des projets sont réussis, 18 % ont échoué et 43 % sont challengés (réalisés avec des ajustements des contraintes) (The Standish Group International, 2013). Le tableau ci-dessous présente les taux de succès des projets de 2004 à 2012.

Tableau 2.1 Résultat d'exécution de projets selon le rapport CHAOS de 2004 à 2012
Tiré et adapté de (The Standish Group International, 2013)

	2004	**2006**	**2008**	**2010**	**2012**
Réussi	29 %	35 %	32 %	37 %	39 %
Échoué	18 %	19 %	24 %	21 %	18 %
Challengé	53 %	46 %	44 %	42 %	43 %

Ce rapport montre une nette augmentation du taux de succès des projets de 2004 à 2012, passant de 29 % à 39 %.

2.2.2 Les facteurs de succès des projets

Depuis les années 2004, la plupart des entreprises de développement logiciel ont adopté les méthodes agiles au détriment des méthodes classiques (Cascade, itérative évolutionnaire, etc.). Aussi, la réalisation des gros projets complexes a laissé place progressivement à celle des petits projets faciles à gérer. Cet état de choses est illustré par les résultats du rapport CHAOS 2013. Ce rapport montre une nette amélioration du taux de succès des projets (de 29 % en 2004 à 39 % en 2012) (The Standish Group International, 2013). Le tableau ci-dessous indique facteurs à prendre en compte pour assurer le succès des projets.

Tableau 2.2 Les facteurs de succès des projets selon le rapport CHAOS 2013
Tiré et adapté de (The Standish Group International, 2013)

Facteurs de succès	**Points**
1. Le soutien de haute direction	20
2. L'implication des utilisateurs	15
3. L'optimisation	15
4. Les ressources qualifiées	13
5. L'expertise en gestion de projet	12
6. Les processus agile	10
7. Les objectifs d'affaires clairs	6
8. La maturité émotionnelle	5
9. L'exécution	3
10. Les outils et infrastructures	1

Ce tableau montre l'importance des processus agile dans le succès des projets. L'implication des utilisateurs (pratique de l'agilité) est aussi un facteur majeur dans le succès des projets.

Certains travaux de recherches ont été effectués afin de déterminer l'impact du choix des méthodes de développement sur le succès des projets de développement logiciel. À ce titre (Khan & Beg, 2013) ont permis de mettre en œuvre un guide de sélection des méthodes de développement (Cascade, Modèle en V, incrémental, XP et RUP) en fonction d'un certain nombre de critères de choix relatifs à la criticité des projets. D'autres recherches ont été menées afin de concevoir un modèle décisionnel du choix du cycle de vie des projets en fonction d'un certain nombre de critères (Khan, Parveen et Sadiq, 2014). Cette recherche a consisté à faire la comparaison par paires de quelques méthodes de développement regroupées en trois catégories (traditionnel, agile et hybride) par la méthode d'analyse AHP (Analytic Hierarchy Process). Ce modèle permet de faire le meilleur choix de méthode de développement en fonction quelques critères de base telle que :

- La définition des requis ;
- Le coût et le temps de développement ;
- La prise en compte des demandes de changements pendant le développement ;
- La complexité du système.

Tous ces travaux de recherche montrent l'importance du choix de la méthode de développement adapté aux exigences d'un projet de développement logiciel et surtout l'importance de l'utilisation des processus agile dans le succès des projets de développement logiciel.

2.3 Les limitations des méthodes agiles

Malgré l'influence positive des méthodes agiles dans le succès des projets de développement logiciel ces dernières années, plusieurs auteurs ont mis en lumière certaines limites pour son utilisation dans le cadre des projets.

Le développement de logiciels complexes : la nature complexe de certains projets de développement logiciels rendent difficiles la gestion des changements et le développement incrémental. Les coûts de gestion des changements deviennent élevés et les projets risquent de dépasser de coût et de délai prévus.

La gestion des sous-traitants : les sous-traitants doivent définir de façon prédictible les besoins et fonctionnalités, le nombre d'itérations pour respecter leur contrat. Or les méthodes agiles recommandent la flexibilité et l'acceptation des demandes de changement durant l'exécution du projet. Plus le nombre de sous-traitants est élevé, plus l'utilisation des méthodes agiles est difficile. Turk, France et Rumpe recommandent à cet effet que les contrats des sous-traitants soient divisés en deux parties. Une partie fixe qui spécifie les exigences et livrables fixes, les activités d'assurance qualité et la méthode d'intégration des demandes de changements. La deuxième partie qui est variable définit les exigences et livrables qui peuvent varier dans les limites définies par la partie fixe (Turk, France et Rumpe, 2005).

L'environnement (équipes de développement distantes) : les communications face à face sont très importantes dans l'agilité. Ce qui est un peu plus difficile à réaliser dans un environnement où les équipes sont géographiquement éloignées. Les outils de communications sont beaucoup plus les documents et les codes (Turk, France et Rumpe, 2005). À cet effet, les documents doivent être mis à jour et centralisés pour faciliter les communications. Quelques auteurs ont abordé cette problématique liée à la gestion des projets de développement logiciel avec des équipes dont les membres sont géographiquement éloignés.

La gestion de grandes équipes : les méthodes agiles sont plus favorables aux petites équipes de développement. Plus les équipes sont grandes, il est difficile de gérer la communication face à face et l'application des principes de l'agilité (Turk, France et Rumpe, 2005), (Hamidi, 2014). L'utilisation des méthodes agiles est recommandée pour les petites équipes. La taille des équipes agiles varie suivant la méthode agile choisie pour la conduite du projet.

La réutilisation des objets : l'une des valeurs de l'agilité est de se consacrer beaucoup plus à la livraison de produit qu'a la documentation excessive. Ceci conduit les équipes de projets de développement à documenter le moins que possible afin de livrer dans les délais le produit alors qu'il est souhaitable et parfois nécessaire d'utiliser des modèles et des objets réutilisables. Malgré la différence entre tous les projets logiciels, il est important pour les entreprises de développement logiciel, de réaliser des objets réutilisables pour plusieurs projets afin d'augmenter la performance des équipes (Turk, France et Rumpe, 2005). Comme recommandé dans le guide du PMBOK, il est important de documenter aussi les leçons apprises et les rétroactions.

2.4 Les avantages du développement de logiciels dans le cloud

Plusieurs travaux de recherches sont consacrés aux avantages de l'utilisation des services du cloud pour le développement logiciel par les méthodes agiles.

2.4.1 Réduction des coûts

Comme évoqué ci-haut (Environnement 3), le développement logiciel dans le nuage offre plusieurs avantages liés aux infrastructures et aux ressources. En effet l'utilisation du cloud réduit les coûts de l'infrastructure de développement et d'implémentation (Kalem et al., 2013a). La mise à l'échelle des applications est plus facile et ne nécessite pas d'énormes coûts d'investissement. C'est le fournisseur de service de nuage qui s'occupe de la gestion et maintenance de l'infrastructure (Tuli, Hasteer, Sharma, & Bansal, 2014), (Mani, Deebitha.S, Jayakumar, & Gopalakrishnan, 2014).

2.4.2 Réduction du temps de développement

Certains travaux de recherches ont évoqué une diminution du temps de développement agile des logiciels lorsque le cloud est utilisé (Kalem et al., 2013a), (Mani et al., 2014), (Abhishek et al., 2012). Ce dernier évoque une réduction de 75% du temps du cycle de développement. Le temps de mise en marché des applications est également réduit (Kalem et al., 2013a). Le déploiement des applications se fait plus rapidement.

2.4.3 Rétroactions des utilisateurs plus rapides

La satisfaction des utilisateurs constitue un des principes clés de l'agilité. À cet effet, les rétroactions des utilisateurs sont très importantes. Avec l'utilisation de l'informatique pour le développement agile des applications, les rétroactions des utilisations sont rapides et ceci facilite donc la prise en compte des demandes de changement (Kalem et al., 2013a), (Cocco et al., 2012).

2.5 Le développement d'applications dans le cloud

Vu les nombreux avantages dans l'utilisation des services de l'informatique en nuage pour le développement logiciel, plusieurs auteurs ont proposé des méthodes, modèles ou guides pour son implémentation concrète :

- Afin de faciliter la migration du développement agile dans un environnement du cloud, (Mwansa & Mnkandla, 2014) propose un guide pour les SMMEs (petites, moyennes et micro entreprises) sud-africaines. Ce guide aborde les points suivants :
    - La détermination des facteurs à maîtriser pour faciliter la migration dans l'environnement du cloud;
    - La détermination des interactions entre ces facteurs dans le succès de cette opération de migration ;
    - La proposition d'une ligne directrice pour la migration du développement agile dans un environnement du cloud;
        - Facteurs importants ;
        - Gestion des processus de migration ;
        - Définition des rôles des parties prenantes internes et externes qui interviennent dans le processus.

- Dans le cadre de la migration, des développements logiciels dans le cloud (Soojin, Mansoo, Sangeun, & Park, 2015) ont proposé un modèle de processus de développement SaaS : SCoDP (SaaS Cloud Oriented Development Process). Ce modèle sert de guide pour le développement de logiciels orienté SaaS. Il a été conçu à partir d'une enquête sur les meilleures pratiques en matière de développement logiciel, en tenant compte des risques, challenges et facteurs de succès.

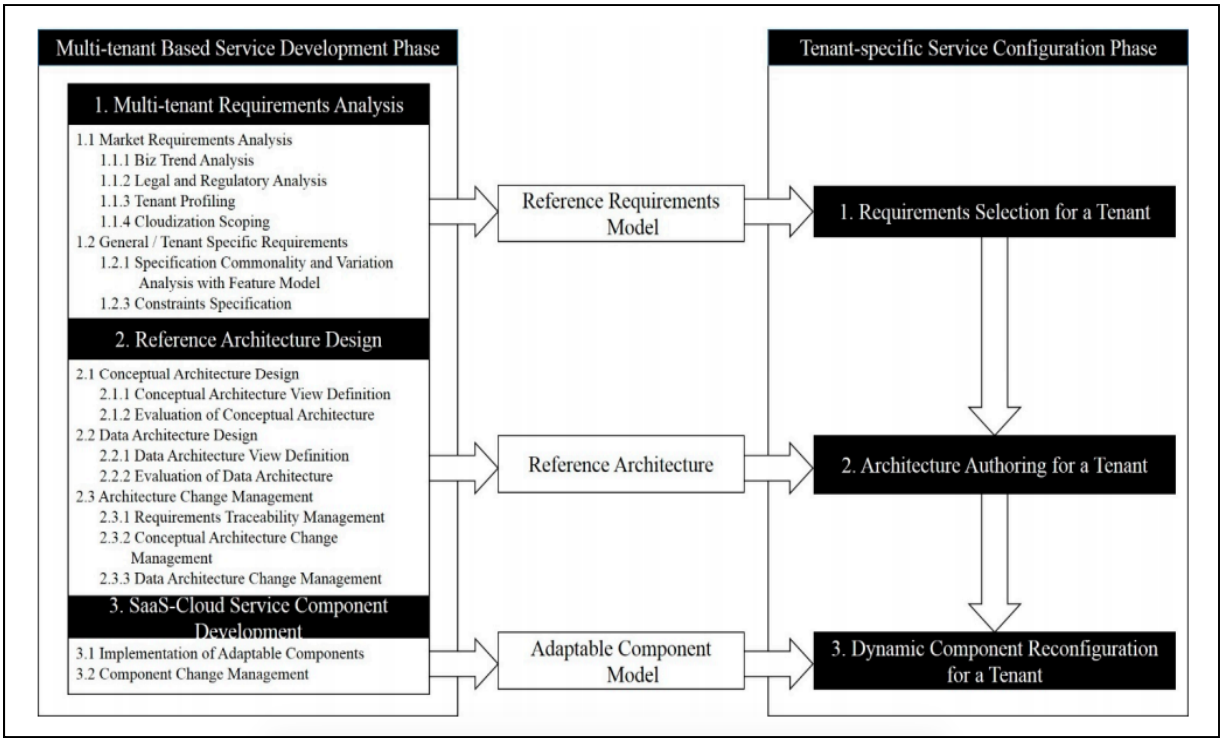

Figure 2.1 Modèle de processus de développement orienté SaaS (SCoDP)
Tirée de (Soojin et al., 2015)

- (Patidar et al., 2011) a proposé une méthode agile qui intègre un rôle supplémentaire (Cloud Provider) aux différentes étapes clés (planification, conception, développement et test) du cycle de développement agile. Ainsi, les activités du fournisseur sont donc définies à chaque étape afin de faciliter la collaboration avec les ingénieurs logiciels. Le succès de cette méthode nécessite une bonne communication et collaboration entre le fournisseur et l'équipe de développement. La figure ci-dessous présente cette méthode.

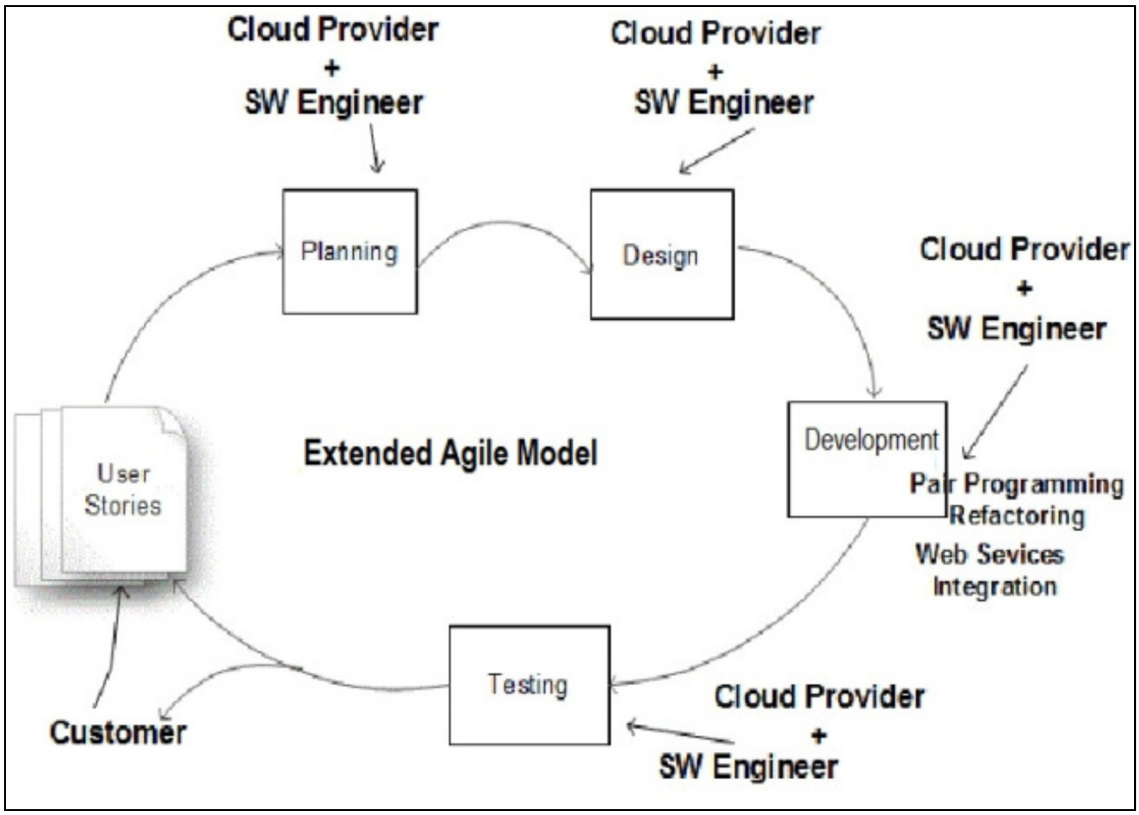

Figure 2.2 Méthode agile étendue
Tirée de (Patidar et al., 2011)

- Le développement de logiciels offshores (logiciels développés par des équipes situées sur des sites géographiquement éloignés) est confronté à plusieurs challenges. (Hashmi et al., 2011), (Chung & Shao-Zong, 2013) ont proposé l'utilisation de la plate-forme de l'informatique en nuage (PaaS) pour le développement de ces logiciels. Ainsi (Cocco et al., 2012) propose une extension de la méthode Agile-Srcum pour l'optimisation du processus de développement d'applications offshores en utilisant la méthode de Système Dynamique de Jaw W. Forrester. La figure ci-dessous présente cette méthode Scrum. Elle met l'accent sur les erreurs et les délais qui sont les facteurs majeurs qui influencent la livraison des fonctionnalités. Ainsi, il propose la création de rôles pour la gestion des délais et faciliter l'optimisation.

- L'un des travaux majeurs sur l'utilisation du cloud pour le développement agile de logiciels est celui de (Singhal, Sonia, & Singhal, 2016). Ces derniers ont proposé la méthode ESCAM (Extended Scrum Cloud Agile Method). Cette méthode est une extension de la méthode Srcum en intégrant les services de l'informatique dans chacune de ses cinq phases (Initialisation, Planification et estimation, Codage et test, Revue et rétrospection), conformément au SBOK (Scrum Body of Knowledge).

2.6 Les tests d'applications dans le cloud

Durant le cycle de développement des logiciels, les tests constituent une étape importante. Il permet de livrer des fonctionnalités de qualité supérieure aux utilisateurs en répondant à leur besoin. En mode agile les tests se font de façon continue dans chaque itération (depuis la de planification jusqu'à la livraison) (Raj, Yadav, & Jaiswal, 2015). Ceci dénote de l'importance des testeurs et de leur implication et collaboration avec les développeurs (surtout pour les projets agiles dont les équipes sont géographiquement distribuées).

Dans le but de réduire la complexité des tests, les coûts, et les délais, plusieurs auteurs ont donc proposé l'utilisation des services de l'informatique en nuage. En effet, en utilisant l'informatique en nuage, les équipes de développement peuvent facilement déployer et configurer les environnements de tests. Elle dispose également des outils pour faciliter les communications(Cruzes, Moe, & Dybå, 2016). À travers leurs travaux de recherche (Raj et al., 2015) propose l'utilisation du TaaS (Testing as a Service) pour l'amélioration des processus de la méthode agile Scrum. Le modèle TaaS se présente comme suit :

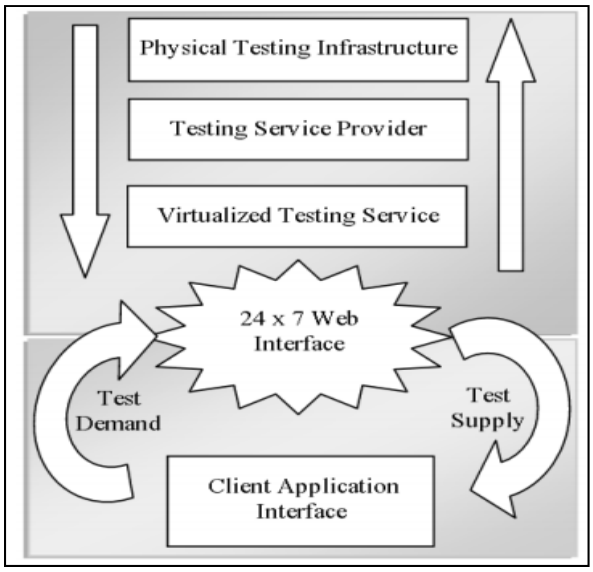

Figure 2.3 Modèle TaaS
Tirée de (Raj et al., 2015)

Ce modèle met en exergue non seulement les tests logiciels, mais aussi les tests des services du fournisseur au niveau de chacune des couches (IaaS, PaaS et SaaS).

C'est dans cet ordre d'idée que d'autres auteurs ont proposé le modèle TSaaS (Test-Support as a Service) pour faciliter et standardiser les tests dans le cloud (Akerele, Ramachandran, & Dixon, 2013; Chana & Chawla, 2013). Les tests sont classés en trois catégories et plusieurs types sont énumérés comme l'indique la figure ci-dessous :

Figure 2.4 Classification des tests dans le cloud
Tirée de (Akerele et al., 2013)

Les différents types de tests sont proposés à travers des interfaces web par les fournisseurs de services au niveau des trois couches que sont Cloud Testing Software-as-a-Service (CTSaaS), Cloud Testing Platform-as-a-Service (CTPaaS) et Cloud Testing Infrastructure-as-a-Service (CTIaaS).

Enfin, pour livrer le plus rapidement les logiciels de qualité, les équipes de développement adoptent les pratiques de DevOps (Mehrotra, 2015). C'est ainsi que Sumit Mehrotra propose donc l'adoption de modèle 4Cs (**Configurabilité, Cohérence, Collaboration et Contrôle**) par les équipes de développement. Ce modèle 4Cs permet d'évaluer les outils et conditions à implémenter sur le cycle de développement agile pour livrer plus rapidement des produits de qualité supérieure. Pour adapter le cycle de développement agile aux contextes du cloud, quatre transformations sont nécessaires (chaque transformation est analysée suivant les quatre conditions du 4Cs) :

- **Transformation 1 :** environnements de test sur demande ;
- **Transformation 2 :** intégration continue ;
- **Transformation 3 :** test à l'échelle de production ;
- **Transformation 4 :** test en parallèle.

2.6.1 Conclusion du chapitre

Dans ce chapitre, il a été question d'analyser les travaux effectués sur l'utilisation des services du cloud pour le développement d'applications agiles.

À cet effet, après avoir présenté quelques limitations des méthodes agiles notamment les difficultés d'utilisation des méthodes agiles sur les projets GSD, il a été observé que la plupart des auteurs ont principalement travaillé sur les avantages et bénéfices de l'utilisation conjointe des méthodes agiles et du cloud pour le développement rapide des logiciels. Ainsi (Abhishek et al., 2012) ont évoqué une réduction temps du cycle de développement de 75 %. Néanmoins d'autres auteurs se sont consacrés aux développements des applications agiles dans le cloud en

proposant des méthodes basées sur quelques méthodes agiles classiques comme Scrum, XP ou DSDM.

En dernier lieu, quelques auteurs ont travaillé sur les transformations à apporter aux méthodes agiles pour réussir les tests dans le cloud. C'est le cas de (Mehrotra, 2015) qui a proposé le modèle 4Cs pour faciliter les tests dans le cloud afin de garantir la livraison rapide et fréquente de logiciels. Dans la suite du mémoire, il est question de présenter les conditions générales d'applicabilité des méthodes agiles dans le cloud afin de minimiser leurs limitations et tirer profit des avantages du nuage.

CHAPITRE 3

L'APPROCHE DEVOPS

3.1 Introduction

Ce chapitre est consacré à la description de l'approche DevOps, ses pratiques et outils qui permettent de livrer plus rapidement et fréquemment des applications et services. Après avoir défini ses caractéristiques et avantages, il est question de présenter les pratiques et les outils nécessaires à sa mise en œuvre. Vers la fin du chapitre l'approche DevOps est abordée dans un contexte de cloud computing.

3.2 Présentation

DevOps est une approche basée sur les principes Lean et Agile et qui regroupe l'ensemble des parties prenantes (développement, assurance qualité, exploitation) collaborant ensemble pour délivrer des logiciels en continu. Ceci permet donc aux entreprises de saisir plus rapidement les opportunités du marché et d'accélérer la prise en compte des retours clients. Également, elle permet de rapprocher les perspectives antagonistes en aidant les équipes à définir ensemble les objectifs métier et à les modifier en continu en fonction des retours de manière à améliorer à la fois l'agilité et les résultats d'entreprise (Coyne, 2015). DevOps intervient donc sur l'ensemble du cycle de vie des applications.

3.2.1 Avantages

Au lieu d'être simplement une fonctionnalité informatique, Devops améliore la manière dont les entreprises apportent de la valeur à leurs clients, fournisseurs et partenaires. Elle offre les avantages sur les 3 domaines ci-dessous (Coyne, 2015) :

- **Amélioration de l'expérience client** : permet de fidéliser les clients et d'accroître les parts de marché. Pour apporter cette expérience, une entreprise doit obtenir et répondre en

continu aux retours clients, ce qui nécessite des mécanismes pour recevoir ces informations de toutes les parties prenantes impliquées dans l'application logicielle à délivrer.

- **Accroissement de la capacité à innover** : DevOps applique les pratiques Lean afin de réduire le gaspillage et la reprise de travail tout en exécutant les activités à forte valeur ajoutée.

- **Accélération du retour sur investissement :** implique de développer une culture, des pratiques et une automatisation afin de délivrer les logiciels rapidement, efficacement et de manière fiable jusqu'à la mise en production.

3.2.2 Fonctionnement

En dépit des principes de l'agilité et Lean, l'approche DevOps s'appuie essentiellement sur 4 principes (Coyne, 2015). Ces principes sont basés sur le concept de « **shift left** ». Ce concept illustré dans la figure ci-dessous vise à permettre aux équipes de développer et tester les applications dans des environnements semblables à l'environnement de production.

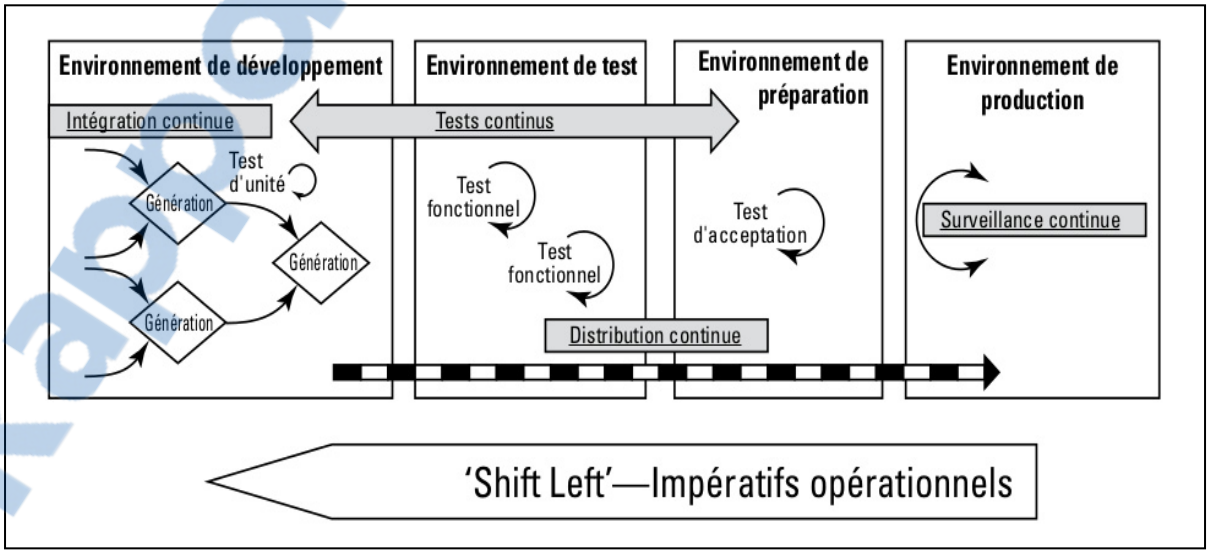

Figure 3.1Le concept " Shift left "
Tirée de (Coyne, 2015)

Cette figure montre aussi les différents environnements et pratiques clés de DevOps. Les principes sont les suivants :

- Développement et test sur des systèmes similaires à ceux de la production « **shift left** »;
- déploiement avec des processus réutilisables et fiables ;
- surveillance et validation de la qualité opérationnelle ;
- amplification des boucles de retour.

3.3 Architecture

L'architecture proposée par Coyne fournit les fonctionnalités nécessaires à l'adoption complète de l'approche DevOps. Ces fonctionnalités s'appuient en grande partie sur les équipes, les pratiques et les outils d'automatisation.

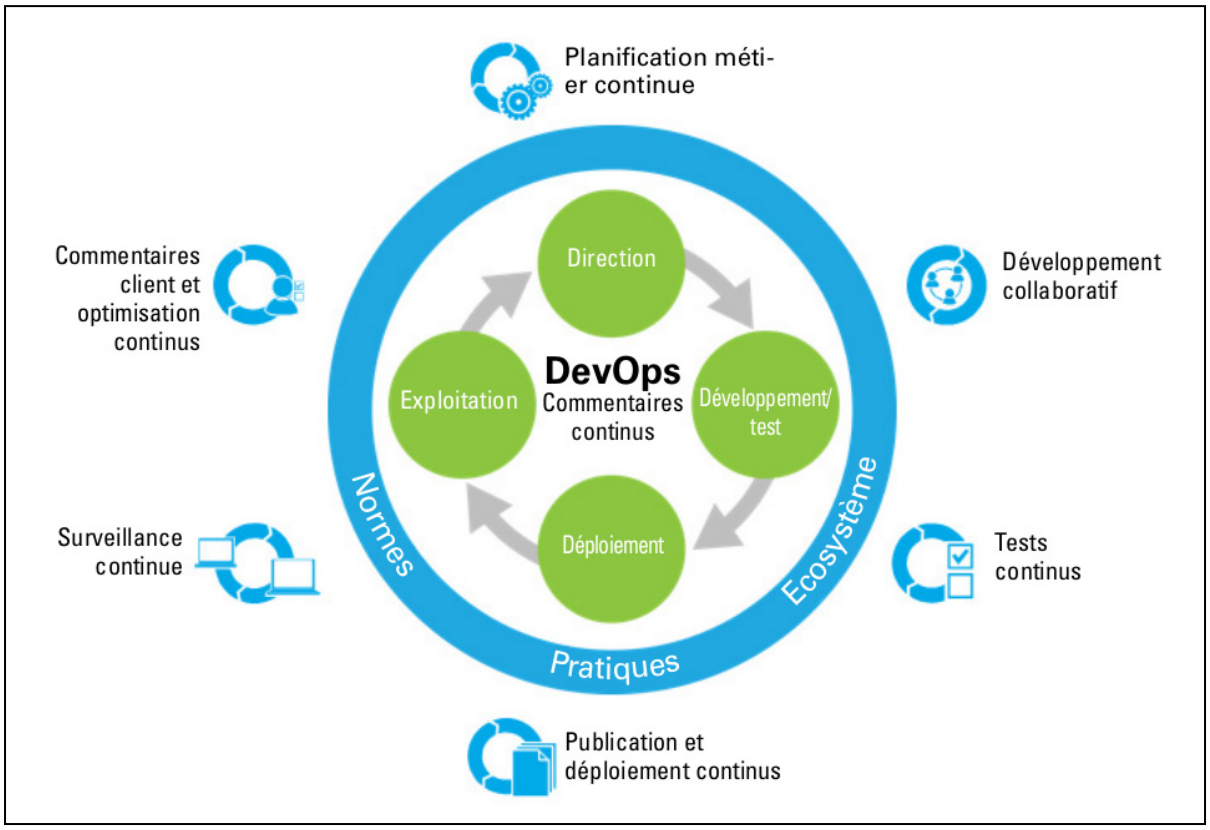

Figure 3.2Architecture DevOps
Tirée de (Coyne, 2015)

Cette architecture est structurée autour de 4 champs d'action qui définissent des pratiques et des normes dans un écosystème bien déterminé :

1. **Direction** : permet de rendre les équipes plus agiles en éliminant le mode de travail en silos. Ce qui amène ces équipes à se focaliser sur les objectifs métiers et réduire et éliminer les délais de livraison.

2. **Développement/test** : permet d'adopter un certain nombre de pratiques qui facilite le développement collaboratif et les tests continus.

3. **Déploiement** : est l'une des champs d'actions clés de l'approche DevOps. Il permet de créer le pipeline de distribution nécessaire à la livraison et au déploiement continu des applications dans les différents environnements.

4. **Exploitation** : permet de surveiller le fonctionnement des applications et gérer de façon continue les retours clients.

3.4 Les pratiques de DevOps

Plusieurs pratiques sont proposées pour l'adoption de l'approche DevOps. Quelques pratiques clés sont illustrées par le tableau ci-dessous :

Tableau 3.1 Liste de pratiques DevOps

	Pratiques
1.	Amélioration continue
2.	Planification des versions
3.	Intégration continue
4.	Développement collaboratif
5.	Livraison continue
6.	Tests continus
7.	Surveillance et retours continus

La liste n'est pas exhaustive, car d'autres pratiques ont été ajoutées en raison de l'utilisation du cloud pour le développement des applications.

3.5 DevOps et le cloud computing

3.5.1 Apports du cloud

Comme nous l'avons évoqué ci-haut, DevOps a pour principal objectif de réduire les goulots d'étranglement dans le pipeline de distribution pour le rendre plus efficace. Parfois les demandes en besoins d'environnements de développement, de test ou de production peuvent prendre plusieurs jours, ce qui occasionne des délais de livraison plus longs. A cet effet, vu que les services du cloud sont à la demande et rapide à provisionner, le cloud vient donc apporter :

- La rapidité du provisionnement d'environnement sur les plateformes cloud peut fournir aux utilisateurs un accès d'usage en libre-service avec une disponibilité d'environnement et un accès à la demande ;
- la possibilité de provisionner dynamiquement ces environnements en fonction des besoins améliore la gestion des environnements et diminue les coûts en réduisant la nécessité d'avoir des environnements de tests statiques permanents ;
- la possibilité d'exploiter des technologies qui s'appuient sur des « modèles » et qui donnent la possibilité aux entreprises de définir et de gérer les versions des environnements comme des logiciels ;
- la possibilité de disposer des technologies d'automatisation des déploiements, la disponibilité des technologies d'automatisation du déploiement des applications ;
- la disponibilité des technologies de virtualisation de services afin de faciliter les activités de tests sans avoir à provisionner des instances réelles de ces services.

A part ces apports certains grands fournisseurs de service de cloud comme Amazon et Microsoft proposent les atouts suivants pour l'utilisation de DevOps dans le cloud :

- **Infrastructure en tant que code** : remplacer l'infrastructure traditionnelle par des infrastructures programmables facile à provisionner et gérer ;

- **Miroservices** : décomposition des applications en de services métiers qui communiquent entre eux par des API ;
- **Surveillance** : capturer, analyser et surveiller les données
- **Intégration et livraison continue :** développement, test et déploiement rapide des applications grâce à l'automatisation.

3.5.2 Modèles de services cloud

Selon les travaux de (Coyne, 2015), le choix du modèle de cloud pour le support de l'approche DevOps dépend du niveau de responsabilité que l'entreprise est prête à gérer elle-même. Ainsi, ils proposent l'utilisation de 3 modèles de cloud (**IaaS**, **PaaS ou Hybride**) :

- **Le modèle IaaS** : l'entreprise gère toute le pipeline de distribution DevOps. Cette dernière gère l'acquisition, l'intégration et la mise à jour de l'ensemble des outils de déploiement du pipeline. L'utilisation du modèle IaaS implique donc une bonne compréhension des outils du pipeline de distribution pour l'adoption de DevOps.

- **Le modèle PaaS** : ici, l'entreprise est responsable uniquement l'application et des données. Elle ne gère donc pas les outils du pipeline de distribution. Ces outils sont à la charge du fournisseur de service cloud. Ce qui permet aux équipes de se concentrer principalement sur les tâches de déploiement des applications. Ces outils sont proposés comme des services à la demande pour chacune des composantes du pipeline.

- **Le modèle hybride** : ce modèle revient à adopter l'approche DevOps dans un environnement de cloud hybride. Le pipeline est partagé entre l'infrastructure cloud et l'infrastructure physique. Les environnements n'étant pas toujours les mêmes, ceci rend plus complexe l'application l'approche de DevOps.

3.6 Conclusion du chapitre

DevOps regroupe l'ensemble des équipes, des processus et des produits nécessaires à une livraison continue de valeur aux utilisateurs finaux. DevOps qui est une extension de l'agilité est de plus en plus adopté dans le cloud pour le développement des applications à cause de ses nombreux avantages. Trois modèles d'adoption dans le cloud sont proposés (PaaS, IaaS et hybride). Avec le modèle d'adoption PaaS de DevOps les équipes sont plus focalisées sur la livraison de l'application. Le prochain chapitre analyse de façon générale le processus de développement dans le cloud.

CHAPITRE 4

DÉVELOPEMENT D'APPLICATIONS DANS LE CLOUD

4.1 Introduction

Après avoir analysé d'autres travaux de recherche, il est question ici d'analyser les contours du développement d'applications dans le cloud. Ainsi, dans un premier temps, il s'agit d'analyser les processus de développement d'applications en fonction des environnements ; puis à partir de la norme ISO/IEC 17789:2014, déterminer les différentes composantes du développement d'applications dans le cloud. À la fin, le développement d'applications est analysé suivant chacune des couches (IaaS, PaaS et SaaS) du cloud avant que les caractéristiques essentielles d'un processus de développement dans le cloud ne soient présentées.

4.2 Les processus de développement d'applications

Le développement logiciel se fait en plusieurs phases ou étapes. Ces phases sont réalisées dans plusieurs environnements en fonction des besoins et contraintes des bénéficiaires (clients, entreprises, utilisateurs) et des exigences technologiques. Les figures ci-dessous présentent trois modèles de développement logiciels basés sur leur environnement de développement et d'implémentation.

Environnement 1

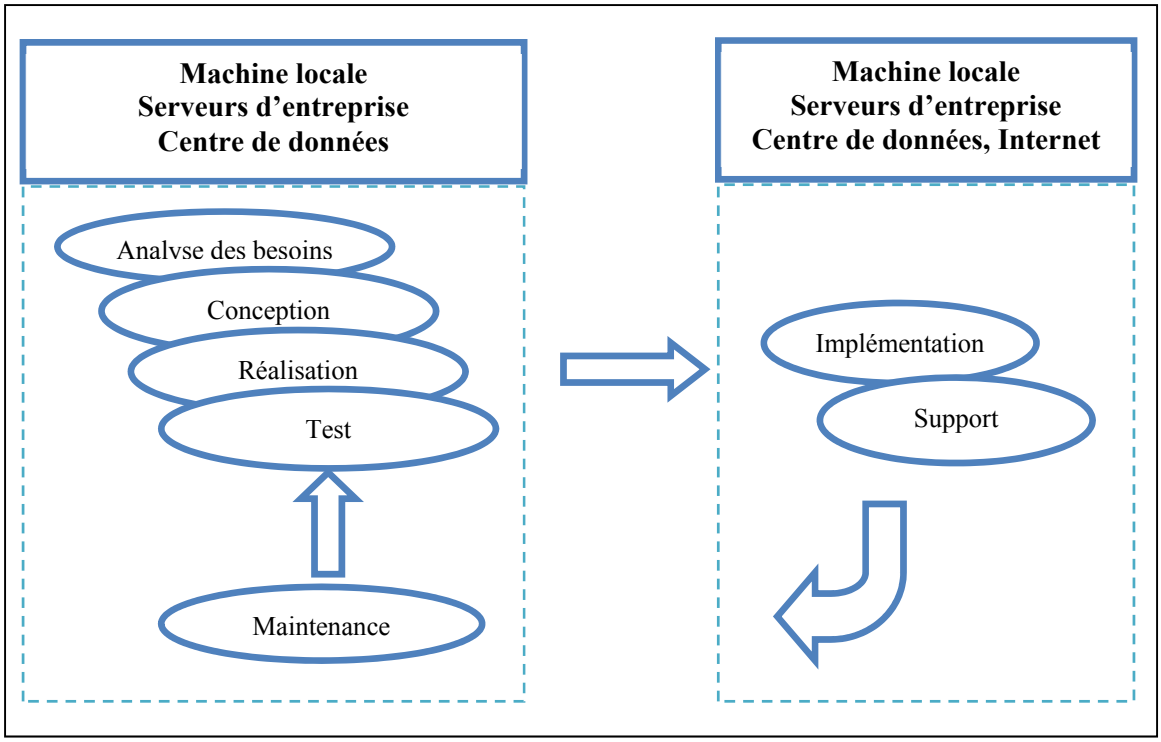

Figure 4.1 Environnement de développement logiciel 1

L'environnement de développement est différent de celui de déploiement et d'utilisation des logiciels. Les logiciels sont développés sur des machines locales, soit sur des serveurs d'entreprise ou soit dans des centres de données d'entreprise dédiés. Après la réalisation et les tests, les logiciels sont implémentés sur l'infrastructure de l'entreprise bénéficiaire ou sur internet pour que les utilisateurs puissent les télécharger. La maintenance est assurée dans l'environnement de développement puis redéployer après les corrections ou la création de nouvelle version.

Ce modèle de développement dans un contexte agile a quelques lacunes selon (Kalem et al., 2013a). En effet, dans leur travail de recherche qui a consisté à comparer le mode de développement logiciel agile sans cloud et celui avec le cloud. Le délai entre la cession des livrables et leur utilisation par le client est très long. Il faut refaire certains tests au niveau

l'environnement d'implémentation et le coût de maintenance de ces deux environnements est assez élevés ainsi que le temps de livraison à chaque itération (Kalem et al., 2013a).

Environnement 2

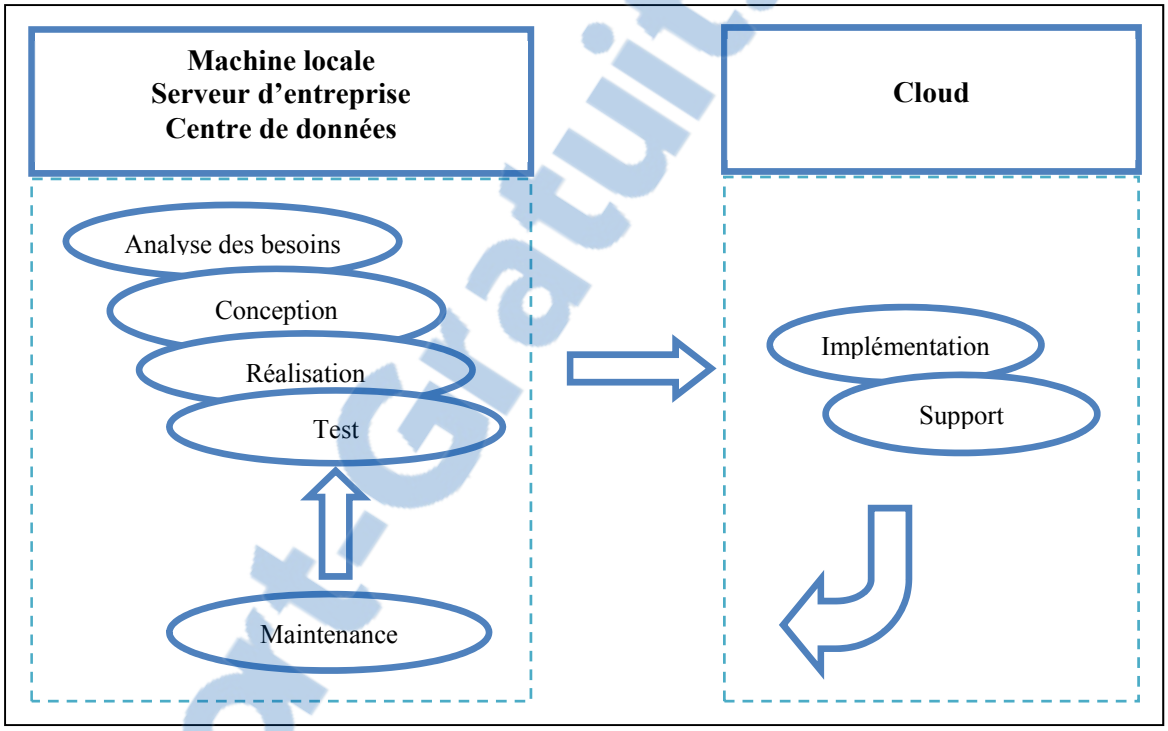

Figure 4.2 Environnement de développement logiciel 2

Dans ce modèle de développement, les environnements de développement et d'implémentation sont aussi séparés. Les logiciels sont développés sur des machines locales, soit sur des serveurs d'entreprise ou soit dans des centres de données d'entreprise dédiés. Mais ici, après la réalisation et les tests, les logiciels sont implémentés dans le cloud, notamment en tant que SaaS. Les logiciels et bases de données sont déployés dans le cloud et sont accessibles aux utilisateurs sur internet en tout instant par le biais des navigateurs web et autres outils web dédiés. La maintenance est plus facile et est transparente aux utilisateurs.

Ce modèle a pour avantage de réduire le coût de maintenance des environnements de production qui est souvent un service à la charge du CSP. Le PaaS est souvent utilisé pour le déploiement et la mise en production des logiciels (Kalem et al., 2013a). Seule l'infrastructure de développement est maintenue par les entreprises ou équipes de développement logiciel.

Environnement 3

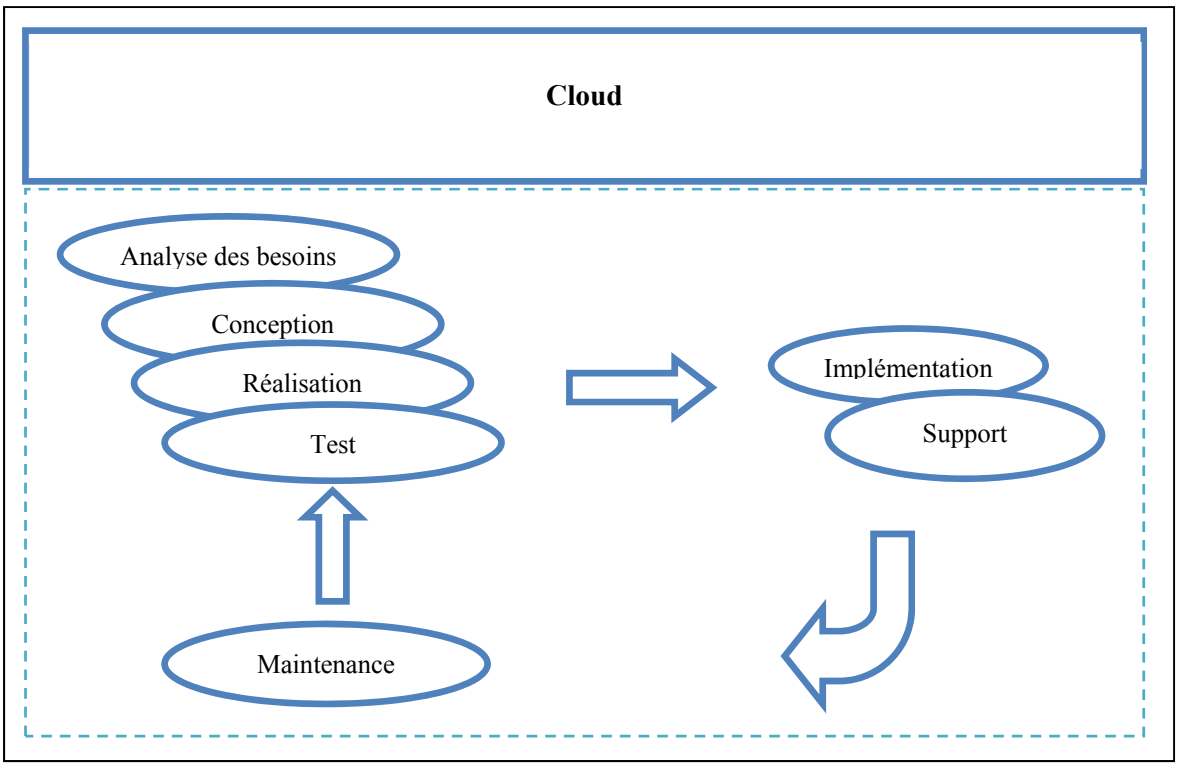

Figure 4.3 Environnement de développement logiciel 3

Pour ce dernier cas, les environnements de développement et d'implémentation sont les mêmes. Les logiciels sont développés, testés et implémentés dans le cloud. Tout le processus de développement est lié aux services du cloud. Les logiciels sont développés, testés et déployés de façon continue directement dans le cloud, ce qui facilite la maintenance et le processus d'amélioration continue.

Dans un contexte de développement agile, les travaux de (Abhishek et al., 2012) ont montrés que l'utilisation des services du cloud peut réduire de 75 % le temps de cycle de développement agile (Abhishek et al., 2012; Jain & Dubey, 2014). On note aussi une réduction du temps de mise en marché des logiciels (Kalem et al., 2013a). D'autres avantages sont liés à l'utilisation de l'infrastructure et des ressources du cloud.

4.3 Le développement d'applications dans le cloud (norme ISO/IEC 17789)

La norme ISO/IEC 177789 décrit l'ensemble des composantes fonctionnelles définissant les relations entre les fournisseurs de services du cloud et les développeurs d'applications.

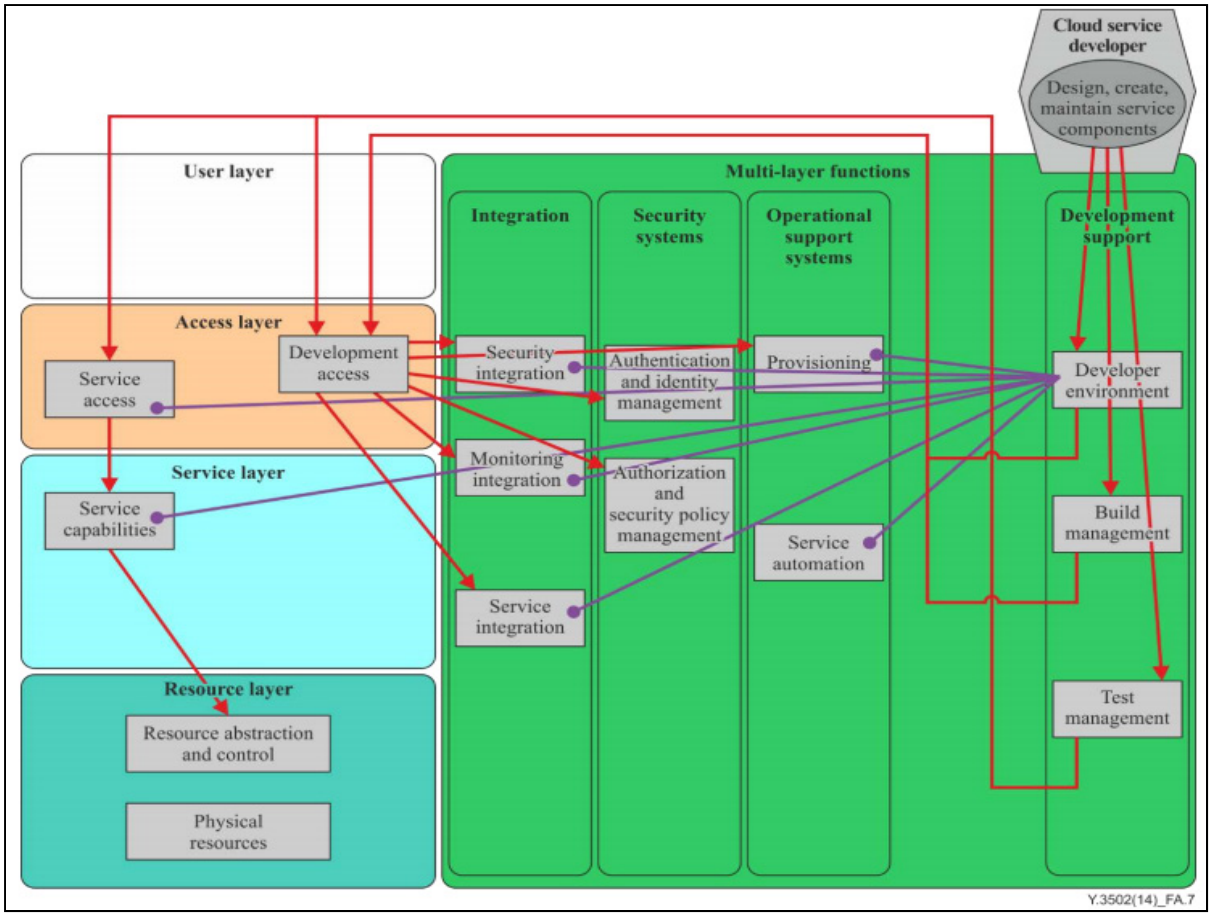

Figure 4.4 La relation entre le « cloud service provider » et le « cloud service developer »
Tirée de (ISO/IEC, 2014b)

La figure ci-dessus présente les services et fonctions fournis par les « cloud service provider » aux développeurs pour la réalisation des applications dans le cloud. Ainsi, les activités des développeurs se regroupent en quatre grandes catégories :

- **Le développement d'applications** : il concerne toutes les fonctions, outils et services nécessaires à la création, la composition, l'exécution et tests des applications dans le cloud. Les environnements sont mis à disposition des développeurs pour assurer cette fonction :

    - Les outils de développements ;
    - Les outils de test ;
    - Les outils de gestion des versions ;
    - Les outils de contrôle de versions.

- **La livraison d'applications** : constitue l'ensemble des fonctions et outils nécessaires à l'exécution des applications dans le cloud.

    - Les services d'intégration ;
    - Les services de déploiement ;
    - Les services d'exécution ;
    - Les services de données.

- **Le contrôle des accès, des services et des ressources :** il concerne les fonctions et outils utilisés pour contrôle des accès aux services et ressources nécessaires au développement des applications dans le cloud.

    - Les services de contrôle d'accès ;
    - Les services de monitoring.

- **La gestion des processus de développement :** En plus des trois catégories précédentes, plusieurs fournisseurs proposent des fonctions et services pour la gestion des processus de

développement dans le cloud. La plupart de ces processus de développement sont basés sur les méthodes agiles.

4.4 Le développement d'applications au niveau des 3 couches du cloud

Le développement d'applications dans le cloud concerne en général toutes les couches de services du cloud.

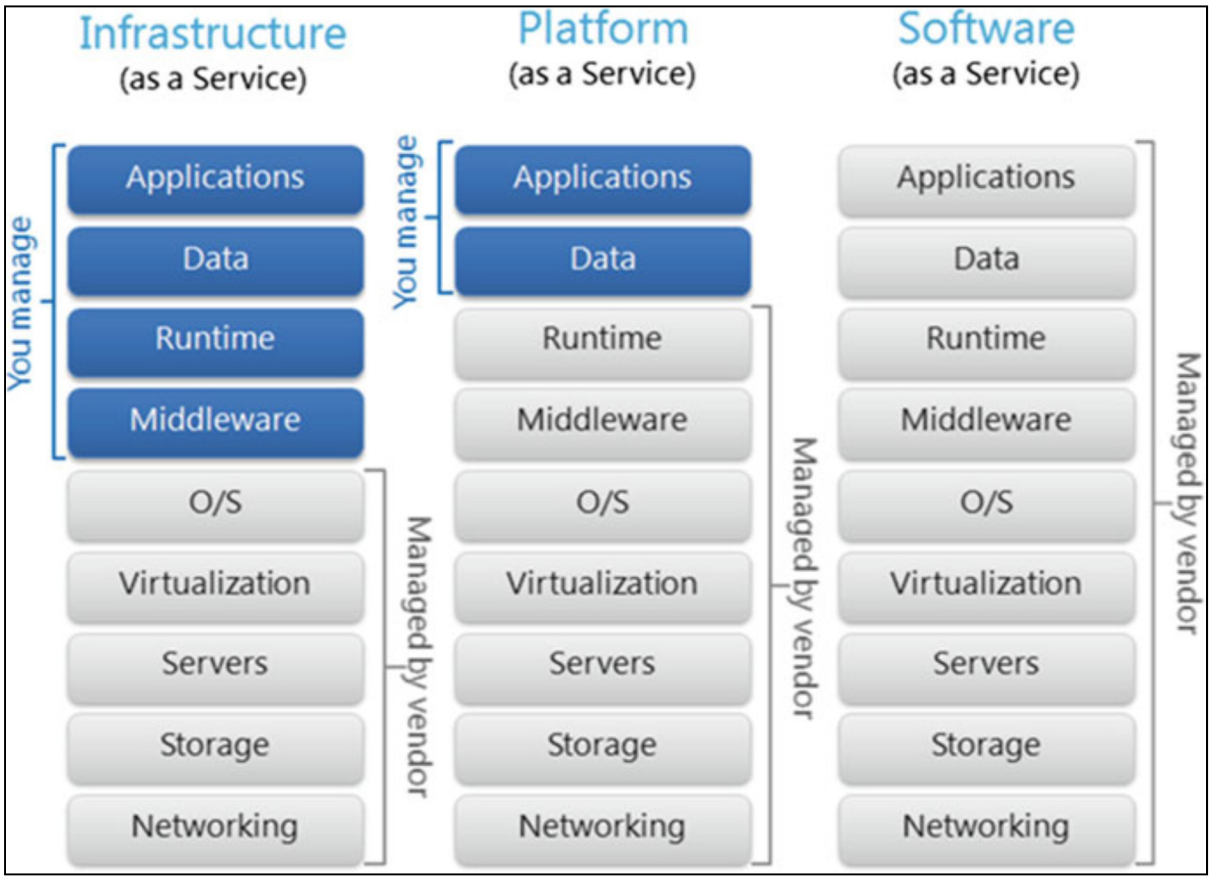

Figure 4.5 Organisation des ressources du Cloud par couche de services
Tirée de (Pramod, Muppalla, & Srinivasa, 2013)

IaaS : elle permet l'accès aux ressources comme les systèmes d'exploitation, les stockages de données, et le réseau. Elle permet donc le développement des logiciels critiques qui nécessitent des configurations de haut niveau concernant les infrastructures. Elle répond aux exigences

volatiles des besoins en infrastructure pour un projet de développement logiciel. La gestion des changements est aussi facilitée dans le cadre d'un projet agile. Les environnements, et systèmes d'exploitation peuvent être mis à l'échelle facilement lors du développement d'applications dans le cloud. Son utilisation pour le développement d'application exige donc plus de travail et d'investissement financier. La maintenance de certaines applications, systèmes d'exploitation et d'autres ressources sont à la charge du client ou de l'équipe de développement.

PaaS : Cette couche offre les ressources logicielles nécessaires à la création, le test et le déploiement des applications. Elle donne accès à un certain nombre de services pour faciliter les activités de collaboration pour le développement d'applications. Elle facilite donc le développement agile d'application dans le cloud et permet aux développeurs de se consacrer principalement aux activités de création des applications sans trop se préoccuper de l'infrastructure sous-jacente qui est gérée par les fournisseurs.

SaaS : cette dernière couche permet de rendre accessibles les applications développées dans le cloud aux utilisateurs. Les fournisseurs de services offrent également aux développeurs des applications en mode SaaS pour faciliter les activités de développement, de collaborations et le monitoring des services offerts. Le développement d'application dans cette couche un peu plus complexe, car il revient à utiliser des applications SaaS paramétrables pour créer d'autres applications SaaS.

Dans ce mémoire, le développement d'application dans le PaaS a été choisi, car elle facilite la création d'applications par l'assemblage de services dont la maintenance est à la charge des fournisseurs de cloud. Aussi, cette couche permet de tirer profit des avantages de l'utilisation du cloud.

4.5 Caractéristiques des processus de développement dans le cloud

Le développement d'application dans le cloud exige une adoption des processus de développement actuels afin d'atteindre la satisfaction des clients. En nous basant sur les travaux de Debbrata Das et Kirti Vaidya intitulés « An agile process framework for cloud application development » (Vaidya, 2011), nous avons identifié quelques caractéristiques essentielles pour les méthodes de gestion des projets de développement d'applications agiles dans le cloud :

4.5.1 Rapide

Le processus doit faciliter le développement rapide des applications par l'assemblage et la composition des services du cloud (Vaidya, 2011). L'utilisation de l'environnement du cloud doit permettre aux équipes de développement de se concentrer sur les activités de développement, de test et de livraison rapide des applications. Ceci favorisera la mise en marché rapide et fréquente des applications.

4.5.2 Configurable

Il doit être configurable facilement afin de répondre aux exigences du projet et de la maturité des équipes de développement. Cette configurabilité permet de l'adapter aux besoins de l'application à développer.

4.5.3 Adaptable et évolutive

Le processus doit être défini de façon flexible pour faciliter l'adaptation des équipes aux différents projets de développement d'applications.

4.5.4 Minimal

Le processus de développement doit inclure le minimum de pratiques, d'artéfacts et de rôles nécessaires à la livraison des fonctionnalités.

4.5.5 Collaborative

L'une des valeurs de l'agilité étant la collaboration, il important que le processus favorise la collaboration au niveau des équipes de développement d'une part et entre les différentes parties prenantes d'autre part.

4.5.6 Fiable

Les services du cloud étant accessibles sur internet par des interfaces web, le processus doit prendre en compte les mécanismes d'authentification et d'accès sécurisé.

4.5.7 Orienté vers le cloud

L'architecture et le fonctionnement des applications dans le cloud étant un peu différents des applications classiques, le processus doit être orienté vers l'utilisation des services et microservices du cloud. En effet, les applications du cloud se basent fondamentalement sur l'assemblage de services fournis par les fournisseurs (cas de PaaS).

4.6 Conclusion du chapitre

Ce chapitre a servi à montrer l'ensemble des composantes nécessaires au développement d'applications dans le cloud. La norme ISO/IEC 17789 a permis d'identifier l'ensemble des activités des développeurs dans le cloud. Parmi ces activités, il y a aussi la gestion des processus de développement qui concerne l'ensemble des activités destinées à faire le suivi des développements. La plupart des fournisseurs proposent des outils basés sur les méthodes agiles. Vers la fin, les caractéristiques d'un bon processus pour de développement dans le cloud

sont identifiées. Dans la suite du mémoire, le développement d'applications se basera sur la couche PaaS et sur la méthode agile Scrum.

CHAPITRE 5

ADAPTABILITÉ DES MÉTHODES AGILES DE DÉVELOPPEMENT DANS LE CLOUD (PAAS) : CAS DE SCRUM

5.1 Introduction

Ce chapitre consiste à présenter et décrire les aspects à prendre en compte pour la gestion des rojets de développement d'applications agiles dans le cloud. À cet effet, la méthode Scrum a été choisie et adaptée pour répondre aux exigences du cloud (PaaS). Pour cette méthode Scrum Cloud, les évènements, rôles et pratiques ont été abordés tout au long de ce chapitre. Notre approche est une approche de gestion et non une approche technique. Ainsi nous ne mettrons pas l'accent sur les activités purement technique du développement dans le cloud.

5.2 Sélection de la méthode Agile Scrum

Depuis ces dernières années, les méthodes agiles sont très fortement utilisées pour le développement logiciel. Comme, il a été montré plus haut, il existe plusieurs méthodes agiles (Scrum, DSDM, Kaban, XP, FDD, etc). Selon le 10^{e} rapport annuel de VersionOne sur l'utilisation des méthodes agiles, la méthode Scrum continue d'être la méthode agile la plus utilisée. En effet, selon l'enquête (VERSIONONE, 2016) 70 % des équipes utilisent Scrum ou les méthodes hybrides Scrum/XP. Selon le 9^{e} rapport de la même institution, la méthode Scrum est utilisée à 56 % (VERSIONONE, 2015), nous notons donc une utilisation de plus en plus accrue de la méthode Scrum par les équipes. Aussi, la méthode Scrum est l'une des méthodes qui répond le mieux aux caractéristiques énoncées dans le chapitre précédent (minimal, adaptative, collaborative et configurable).

Sur cette base, nous avons choisi d'utiliser la méthode Scrum pour l'adapter au contexte de gestion des projets de développement d'applications dans le cloud. Le choix de la méthode Scrum et du modèle PaaS n'est pas original, mais nous permet d'analyser l'applicabilité des méthodes agiles dans le cloud en fonction de ces pratiques, rôles et artéfacts. En effet d'autres

auteurs comme Singhal ont proposé en 2016 la méthode ESCAM qui est une extension de la méthode Scrum dans le cloud.

5.3 Modèle d'utilisation de Scrum

L'un des avantages de la méthode Scrum qui facilite son utilisation est son caractère léger. Elle définit le moins que possible de pratiques, ce qui facilite son adaptation aux besoins des projets et sa combinaison avec d'autres méthodes agiles. Elle peut être utilisée suivant deux modèles :

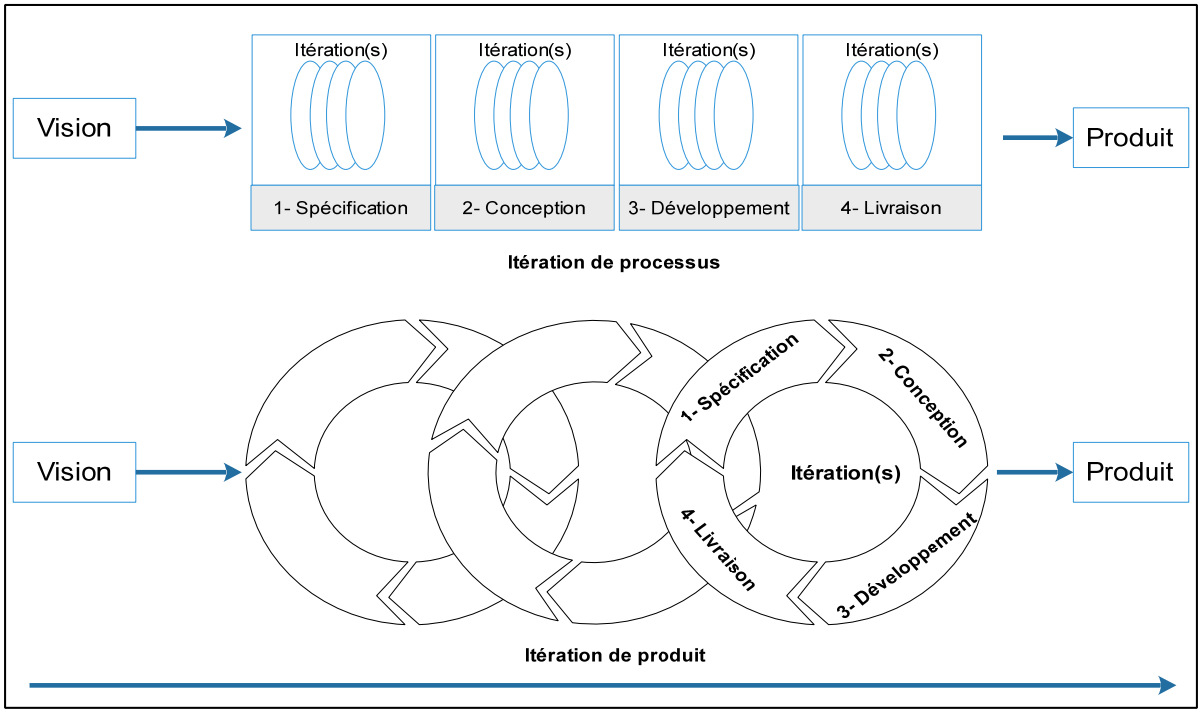

Figure 5.1 Modèle d'utilisation de Scrum

Le premier modèle « **Itération de processus** » consiste à réaliser le projet de développement d'application de façon linéaire et itérative. Chaque phase est réalisée de façon itérative pour livrer des incréments. Néanmoins ces incréments ne permettent pas de vite livrer des scénarios clients qui donnent de la valeur ajoutée au produit après chaque itération.

Par contre, le second modèle « **Itération de produit** » permet livrer plus rapidement des fonctionnalités répondant aux besoins des clients. Au cours de chaque itération, les scénarios

clients sont planifiés, réalisés et livrés. Dans ce modèle, l'équipe est focalisée sur la livraison d'incrément qui donne de la valeur ajoutée au projet. C'est ce modèle qui est adopté dans la suite du mémoire, car il offre plus d'agilité et est amélioré par l'utilisation des services du cloud.

5.4 Méthode Scrum (SBOK)

Le guide du SBOK décrit l'ensemble des rôles, artéfacts, ou pratiques de la méthode Scrum. Il est basé le concept de livraison incrémentale de produit de Scrum. Ce guide est divisé en 3 domaines que sont : Les principes, les aspects et les processus.

5.4.1 Flux de projet Scrum

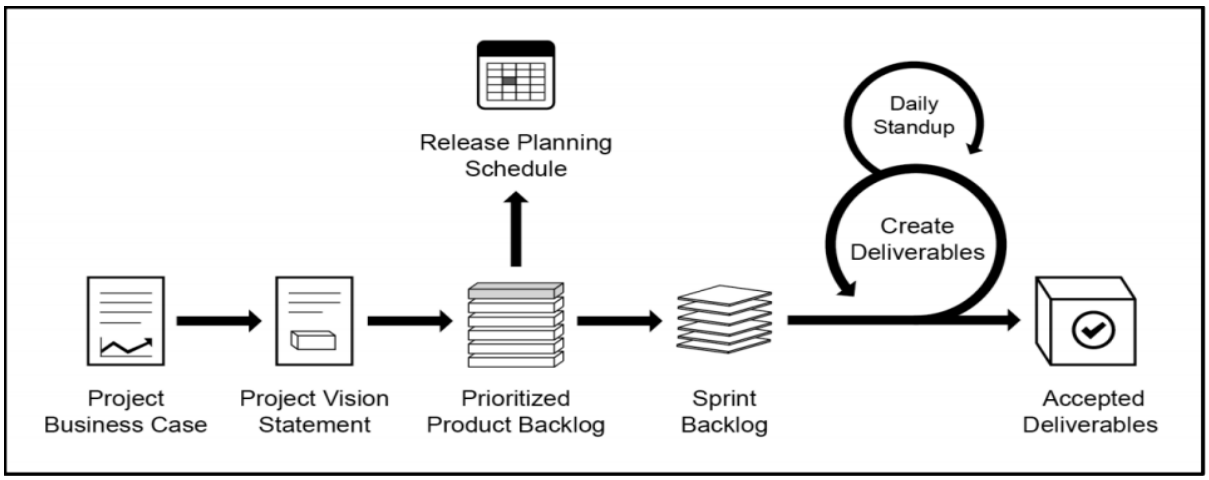

Figure 5.2 Flux d'un projet Scrum pour un Sprint
Tirée de (SCRUMstudy, 2016)

À partir de la vision du projet, les fonctionnalités de l'application à développer sont évaluées et planifiées sur plusieurs blocs de temps de deux à six semaines (**Sprint**). Ces fonctionnalités sont développées de façon itérative et incrémentale dans chaque Sprint puis livrées pour être potentiellement utilisées (SCRUMstudy, 2016).

5.4.2 Pratiques

Tableau 5.1 Les pratiques Scrum (SBOK)
Tiré et adapté de (SCRUMstudy, 2016)

Phase	Pratique Scrum (SBOK)
1. Démarrage	1. Créer la vision du projet 2. Identifier le Scrum Master et le (s) intervenant (s) 3. Former l'équipe Scrum 4. Développer les Epics 5. Créer et prioriser le carnet de commandes de produits 6. Conduite de la planification de la libération
2. Planification et estimation	7. Créer les User Stories 8. Approuver, estimer et valider les User Stories 9. Créer les tâches 10. Estimer les tâches 11. Créer le Backlog de Sprint
3. Développement	12. Créer des livrables 13. Conduite de la mêlée quotidienne 14. Affiner le carnet de commandes
4. Revue et rétrospection	15. Convoquer Scrum de Scrums 16. Démontrer et valider le Sprint 17. Rétrospection du Sprint
5. Livraison	18. Livrer 19. Rétrospection du projet

Ce tableau ci-dessous présente les pratiques Scrum répertoriées par guide SBOK. Le guide Scrum du SBOK présente 19 pratiques réparties en 5 phases. Ces pratiques décrivant l'ensemble des activités du processus sont assurées par l'ensemble des membres de l'équipe Scrum afin d'atteindre les objectifs du projet.

5.5 Adaptation de la méthode Scum au contexte du cloud PaaS

Pour adapter le développement agile d'application au cloud computing, dans ce mémoire il a été choisi la méthode Scrum. Cette dernière basée sur la version 2016 du guide du SBOK, présente l'ensemble des évènements, des pratiques et des rôles nécessaires à la gestion de projet agile dans le cloud. Le développement d'applications pouvant se faire sur les 3 couches (SaaS, IaaS, PaaS) couches fondamentales du cloud, nous nous sommes limités à la couche plate-forme (PaaS) du cloud dans ce projet de recherche. Ainsi, l'implication du PaaS est analysée dans chacun des évènements de développement d'applications abordées.

5.5.1 Méthode Scrum adaptée au PaaS

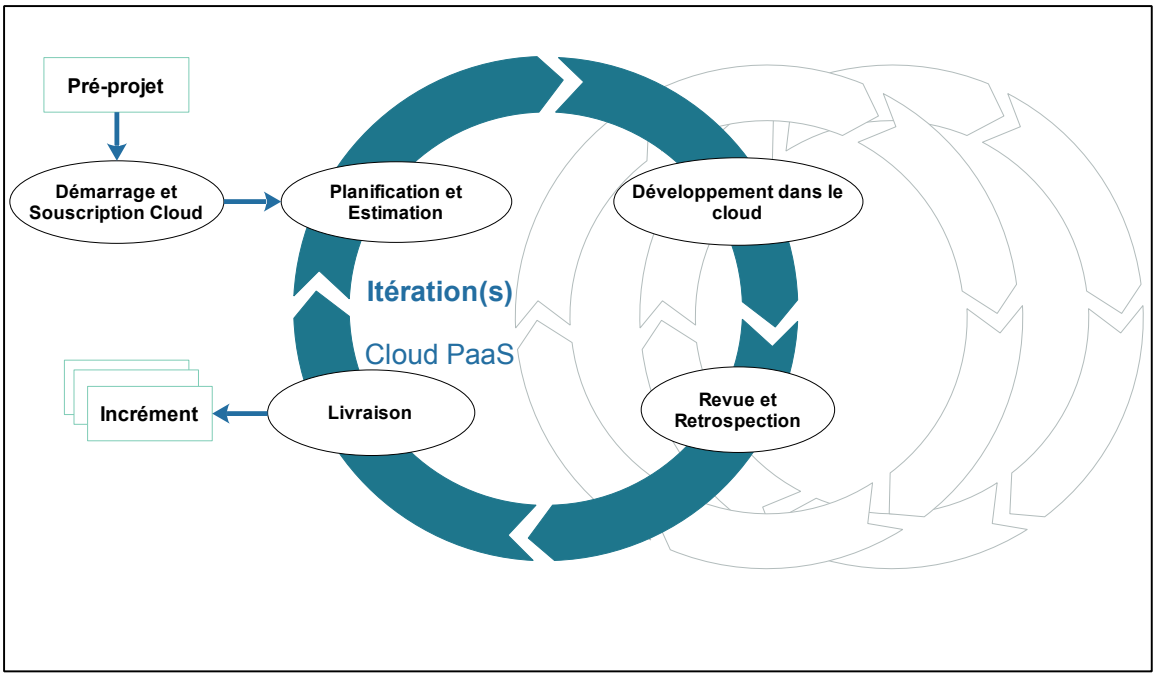

Figure 5.3Méthode Scrum adaptée au PaaS

La figure ci-dessus montre les étapes de la méthode Scrum adaptée au contexte du cloud pour le développement d'applications. Il comporte **six** évènements et qui définit un certain nombre de pratiques qui répondent aux exigences des services du cloud. Cette méthode conserve les principes, rôles et pratiques habituels de la méthode Scrum, mais nous avons identifié et ajouté quelques évènements, pratiques et rôles engendrés par l'utilisation du PaaS dans le processus de développement agile des applications. Dans la suite du travail, les évènements, pratiques et rôles sont présentés en détail.

5.5.2 Description de la méthode

5.5.2.1 Les évènements

Pour étendre la méthode Scrum dans le cloud, les pratiques ci-dessous ont été rajoutées. Les évènements aussi ont subi des modifications. L'évènement de « **Souscription Cloud »** est l'un des plus importants ajoutés pour l'acquisition et la mise à jour continue des services et outils du cloud.

Tableau 5.2 Liste des pratiques Scrum Cloud en fonction des évènements

Évènement	Pratique Scrum Cloud
Pré-projet	1. Analyser les besoins en services cloud 2. Analyser la complexité et la criticité de l'application
Démarrage et Souscription Cloud	3. Évaluer les plates-formes du cloud et choisir les fournisseurs 4. Gérer les contrats et acquérir les services 5. Définir les accès aux services et configuration des environnements
Planification et estimation	6. Planifier les activités
Développement	7. Construire le code et composer les services 8. Effectuer les tests en parallèle 9. Automatiser les tests
Revue et rétrospection	10. Automatiser les rétroactions 11. Auditer les services de cloud 12. Analyser les besoins de mise à l'échelle
Livraison	13. Déployer une version de l'application 14. Traiter les rétroactions des utilisateurs 15. Mettre à l'échelle l'environnement de production

Les évènements de la méthode sont décrits et l'implication de PaaS est évaluée afin d'identifier les aspects à prendre en considérations pour le développement, l'acquisition des plates-formes et leur utilisation. Il faut noter qu'à part cet évènement initial de pré-projet, les autres sont effectués dans des boules d'itérations.

Pré-projet

Il sert à créer la vision du projet et définir les orientations du projet. Cet évènement intervient avant le démarrage des itérations. C'est à cette étape que le **Product Owner** est identifié pour

assurer la livraison des fonctionnalités. Pour le développement d'application dans le PaaS, il est important ici de s'assurer que le choix du cloud est aligné sur la vision de l'entreprise. Il faut donc évaluer les questions de sécurité et de confidentialité de données. Elle connaît la participation de l'ensemble des parties prenantes et des spécialistes du cloud.

Implication du PaaS

Un certain nombre d'aspects sont à prendre en considération afin de choisir les plates-formes PaaS qui répondent aux exigences du projet.

- **Les équipes :** création d'une équipe Scrum multidisciplinaire qui maîtrise les champs du PaaS suivants :

    1. La facturation interne et l'allocation des coûts
    2. L'audit et la gestion des risques
    3. La sécurité et la protection des données
    4. La gestion du changement
    5. La gestion des développements et des opérations

- **Les contrats :** évaluation des « **Cloud Service Agrement (CSA)** » qui définit généralement les 3 aspects suivants :

    1. Les contrats clients
    2. Les politiques d'utilisation acceptables
    3. Le niveau de service requis

    Les fournisseurs de plates-formes PaaS offrent plusieurs fonctionnalités à des prix et des niveaux de service différents et il est important d'en tenir compte afin d'atteindre les objectifs du projet.

- **Les coûts :** évaluation des coûts afin d'identifier s'il existe des limites dans l'acquisition des plates-formes. Cela est relatif aux entreprises qui développent déjà plusieurs projets dans le cloud et qui ont des limites budgétaires.

- **Les besoins d'affaires :** en fonction de la complexité et la criticité de l'application, les plates-formes de PaaS doivent être évaluées pour répondre aux besoins d'affaires.

- **Les licences de logiciels :** certains projets de développement nécessitent l'utilisation de logiciels sous licences. Il faut donc s'assurer que les plates-formes PaaS offrent cette possibilité et faciliteront la mise à l'échelle de ces licences.

- **La sécurité et la disponibilité :** les processus de gestion des identités pour le contrôle des accès aux plates-formes doivent être clairement définis.

- **La gestion des changements :** l'un des avantages des plates-formes PaaS est que les systèmes d'exploitation, les logiciels et API sont gérés par les fournisseurs. Une bonne communication doit être établir entre les fournisseurs et les utilisateurs PaaS afin que ces derniers soient informés à temps des changements majeurs des applications et services.

- **La localisation des données :** certaines applications sont sujettes à des questions de réglementation et d'exigences légales. Il faut donc tenir compte de la localisation géographique des centres de données des plates-formes PaaS.

- **La facturation des systèmes :** il est important de contrôler et de s'assurer que les factures des fournisseurs correspondent aux spécifications techniques des services demandés.

- **Le verrouillage des fournisseurs :** les plates-formes du PaaS offrent beaucoup d'avantages, mais, il faut tenir compte des questions de portabilité et d'interopérabilité pour éviter le verrouillage des fournisseurs. La portabilité vérifie la capacité à déplacer facilement les éléments ci-dessous d'un fournisseur à un autre :

1. Les environnements de développement ;
2. les applications ;
3. les données ;
4. les codes sources.

- **Les processus de sortie du cloud :** s'assurer que les fournisseurs de PaaS proposent aux clients des processus de sortie pour éviter les questions de verrouillage.

Ces aspects concernant la gouvernance et l'utilisation des plates-formes PaaS doivent être pris en compte au début du projet de développement.

Cet évènement n'est pas nouveau dans la méthodologie Scrum, mais nous avons ajouté certaines pratiques supplémentaires qui visent à prendre en compte les spécificités liées à l'utilisation des services du cloud. La prise en compte de ces différents aspects que nous avons cités plus haut permet aux entreprises de s'assurer que les équipes de projets de développement agile comprennent les enjeux du cloud. Aussi, il permet de s'assurer que le projet ne nécessite pas un contrôle de l'infrastructure par l'équipe. En ne tenant pas compte de ces aspects dans cet évènement, le projet de développement risque de ne pas être aligné sur la vision et culture de l'entreprise par conséquent, ne pas satisfaire aux besoins des clients et de différentes parties prenantes. En gros, nous avons identifié les risques suivants :

- Risques de non-alignement stratégique du projet de développement ;
- Risques liés à l'infrastructure, car dans un environnement PaaS, l'infrastructure est n'est pas contrôlé par l'équipe de projet.

Démarrage et souscription cloud

Il sert à identifier le **Scrum Master** et l'ensemble des membres d'équipe de développement du projet. D'autres rôles sont ajoutés aux rôles habituels. Les rôles de **Cloud Broker** et **Cloud**

Auditor sont aussi identifiés. Ces derniers facilitent la sélection des fournisseurs de services de Cloud**.**

Il permet également d'acquérir ou de mettre à jour en fonction des besoins, les services, outils et environnements de développement et de tests pour la livraison des incréments. Les contrats sont négociés avec les fournisseurs de services de cloud. Les accès sont définis pour les membres de l'équipe Scrum.

Implication du PaaS

La plupart des fournisseurs de plates-formes PaaS sont choisis en fonction des paramètres décisionnels montrés ci-haut. En fonction des besoins d'affaires et de la complexité de l'application à développer, une grille d'évaluation et de sélection est établie. La méthode d'analyse AHP peut être utilisée afin d'identifier le ou les fournisseurs qui répondent aux exigences du projet.

En termes d'avantage, cet évènement permet de prendre en compte l'ensemble des contraintes du projet afin de mieux sélectionner les fournisseurs. Une mauvaise sélection des fournisseurs peut entrainer des conséquences sur la qualité du logiciel à réaliser ou sur l'atteinte des objectifs. Ceci est dû au fait que le PaaS offre moins de contrôle que dans les environnements IaaS ou autres environnements classiques de développement. Le changement de fournisseurs de services PaaS en cours de développement peut être préjudiciable au succès du projet avec possibilité de blocage par le fournisseur ou des retards dans la livraison des incréments.

Planification et estimation

En plus des activités classiques d'estimation des activités et de création du **Sprint Backlog,** la gestion des services du PaaS est planifiée. Les environnements et services sont mis à jour au besoin afin de faciliter les activités de développement et de test. Les risques liés à l'utilisation de l'environnement PaaS sont planifiés.

Développement

Il concerne les activités de conception, de création et de composition de l'application correspondant à l'ensemble des fonctionnalités prévues dans le **Sprint**. Plusieurs environnements de développement et de tests sont mis en place pour accélérer les activités. Les tests sont automatisés et effectués en parallèle.
Sur le plan de la gestion, les rencontres quotidiennes sont effectuées afin de suivre l'avancement des activités. Les risques sont identifiés et gérés ainsi que les demandes de changements.

Implication du PaaS

Dans le développement Scrum d'application dans le PaaS, la réalisation d'application revient essentiellement à composer plusieurs services et composants afin de livrer les scénarios clients. De ce fait, l'équipe de développement doit maîtriser l'architecture du PaaS et son fonctionnement. Les outils du PaaS permettent d'assurer la visibilité et la transparence des activités et rendent accessible en tout temps l'évolution de chaque sprint.

Cet évènement permet de spécifier les activités particulières liées au développement dans un environnement PaaS. Ces activités permettent de tirer le maximum de profit de l'utilisation de cet environnement ou l'infrastructure sous-jacente est complètement gérée par les fournisseurs. Cela permet à l'équipe se focaliser beaucoup plus sur les activités de développement et de test.

Revue et rétrospection

Pour cet évènement, les fonctionnalités sont validées par l'ensemble des parties prenantes à partir des critères d'acceptation. Les corrections sont effectuées à cette étape afin d'assurer la qualité de l'application. Les leçons apprises sont documentées et automatisées par les services du cloud dédiés. Elle implique la participation de l'ensemble des parties prenantes et de l'équipe du projet.

Livraison

Ici, une version de l'application est disponible pour la mise en marché. Les leçons apprises sont également documentées et automatisées pour de futures versions de l'application. La configuration des environnements, serveurs de données et de stockage sont faits pour assurer la mise en production de l'application dans le cloud et le rendre disponible en mode SaaS aux utilisateurs. En fonction des besoins du marché, l'application est rendue disponible aux utilisateurs par l'équipe de production.
La livraison rapide est l'une des avantages de l'utilisation du cloud dans le développement les applications. Nous allons donc spécifier plus en détail plus bas l'ensemble des activités liées à la livraison des logiciels dans le cloud.

Synthèse

De façon générale le processus de développement dans le PaaS fait appel à des applications en tant que services pour la livraison de l'application :

- **Développement en tant que service** : l'ensemble des outils et services pour l'édition des codes ou la composition des services ;
- **Construction en tant que service** : pour la construction de l'application ;
- **Déploiement en tant que service** : pour le déploiement de l'application exécutable ;
- **Test en tant que service** : les outils d'automatisation des tests sont fournis pour assurer la qualité de l'application (test de charge, de performance, d'intégration). L'environnement de test sert à restreindre l'accès à une version de l'application à un groupe limité d'utilisateurs ;
- **Monitoring en tant que service** : pour surveiller et mesurer les ressources et services.

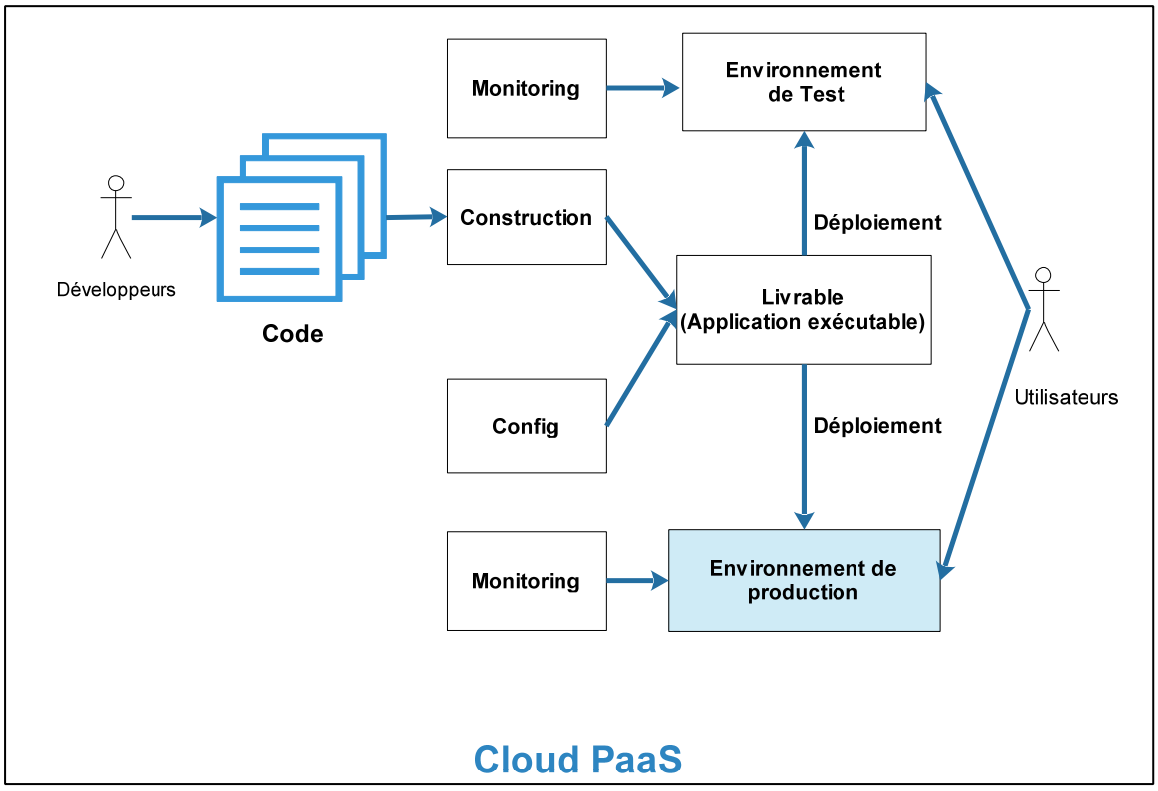

Figure 5.4 Les composantes du développement dans le PaaS

5.5.2.2 Rôles

L'utilisation la méthode Scrum pour le développement d'application dans le cloud implique un certain nombre de rôles. Le tableau ci-dessous indique l'ensemble rôles prévus. Parmi ces rôles deux sont ajoutés en raison de l'utilisation des services du coud.

Tableau 5.3 Liste des rôles Scrum + Cloud

Rôles	Scrum	Scrum + Cloud
1- Product Owner	x	x
2- Scrum Master	x	x
3- Scrum Team	x	x
1. Développeur	x	x
2. Testeur et QA	x	x
3. Architecte	x	x
4. Cloud Broker		x
5. Cloud Auditor		x
4- Parties prenantes	x	x

Le tableau ci-dessous montre que notre méthodologie comporte 2 rôles supplémentaires à la méthode Srum que sont **Cloud Broker** et **Cloud Auditor**.

Rôle 1 : Product Owner

Il assure la livraison de l'application, en gérant les items du **Product Backlog.** Ces items ou « **User Stories** » sont priorisés afin d'assurer le maximum de valeur pour les clients ou l'entreprise. Il utilise donc les outils du cloud pour rendre accessible en tout moment les items du **Product Backlog ou du Sprint Backlog.** Ce rôle ne change pas à part l'utilisation des applications SaaS pour faciliter l'inspection, adaptation et la transparence (qui sont les 3 piliers de la méthode Scrum) des fonctionnalités.

Rôle 2 : Scrum Master

Ce rôle permet de s'assurer que les principes et valeurs de Scrum sont bien respectés au sein de l'équipe. Il joue un rôle de facilitateur pour l'atteinte des objectifs de chaque itération. Dans

le cadre du développement dans le cloud, il s'assure que tous les membres d'équipe maîtrisent l'utilisation des services de cloud pour la livraison de l'incrément.

Rôle 3 : Scrum Team

L'équipe Scrum est une équipe pluridisciplinaire, qui est formée habituellement de **Développeurs**, de **Testeurs**, de **QA** et des **Architectes** qui ont pour rôle principal de créer l'incrément de chaque itération. Mais dans un contexte de développement d'application dans le cloud, il faut aussi les rôles de « **Cloud broker** » et « **Cloud Auditor »**.

Rôle 3.1 : Développeur

Il effectue les activités d'assemblage et d'écriture des codes sources pour la réalisation de l'application. Les développeurs utilisent les environnements de développement, outils et modules fournis par les services du cloud afin de créer les applications.

Rôle 3.2 : Testeur

Ce rôle assure la réalisation des activités de planification et d'automatisation des tests. En collaboration avec les développeurs, ils effectuent les tests fonctionnels, les tests de charge et de performance afin d'assurer la qualité de l'application.

Rôle 3.3 : Architecte

Il propose et travaille en collaboration avec les « **Cloud broker »** et « **Cloud Auditor »** afin mettre de à disposition des développeurs et testeurs, les environnements et ressources nécessaires à la livraison de l'incrément.

Rôle 3.4 : Cloud broker

Ce rôle est ajouté aux rôles habituels de l'équipe Scrum afin de faciliter la relation entre l'équipe de développement et les fournisseurs de service du cloud. Ainsi, il effectue les actions suivantes :

- Aider les équipes de développement dans la sélection des fournisseurs de services ;
- Gérer et négocier les contrats avec les fournisseurs de services ;
- Gérer les demandes de mise à l'échelle des environnements et des services.

Nous avons donc ajouté ce rôle afin de faciliter la gestion des activités citées ci-haut. Il faut rappeler que ce rôle est défini dans la norme ISO 17789 :2014, relatif à l'architecture du cloud. Son intégration aux rôles assurent une bonne compréhension des contraintes et avantages du cloud par l'équipe du projet.

Rôle 3.5 : Cloud Auditor

Il mesure et évalue la performance des environnements et ressources afin de s'assurer que les fournisseurs garantissent le niveau de service requis. Il est très impliqué dans les évènements de démarrage et de souscription des services du cloud.

À l'image du rôle de « Cloud Broker » ce rôle est aussi défini par la norme ISO 17789 :2014. L'inclusion de rôle a pour avantage de contrôler les différents services fournis par les fournisseurs. L'infrastructure étant contrôlée par le fournisseur, il est important de veiller en permanence à la disponibilité des services. L'absence de ce rôle peut entrainer des problèmes d'arrêt des services, de mauvaise gestion des ressources et services ou de non-respect des clauses contractuelles par les fournisseurs en matière que qualité de services.

Néanmoins l'ajout de ces 2 rôles « **Cloud Broker** » et « **Cloud Auditor** » peuvent occasionner une augmentation de l'effectif de l'équipe du projet. En effet la méthode Scrum recommande de petites équipes faciles à maîtriser.

Rôle 4 : Parties prenantes

Ce rôle définit les actions des autres acteurs qui influencent le projet de développement d'application. Il s'agit des clients, des utilisateurs, des commanditaires, etc. Ce rôle ne fait pas partir de l'équipe Scrum.

5.5.2.3 Pratiques Scrum + PaaS

Les pratiques suivantes ont été identifiées pour étendre la méthode Scrum dans le cloud pour le développement et la livraison des applications.

Pratique 1 : Analyser les besoins en services cloud

Le développement d'application dans le cloud étant récent, il est question d'analyser les avantages de l'utilisation des ressources du cloud pour la livraison des applications. Il faut s'assurer que le développement d'application SaaS réponde aux besoins et à la complexité du logiciel. Cette pratique concerne non seulement les dirigeants des entreprises, le **Product Owner**, mais aussi des experts en cloud. Ces derniers faciliteront l'analyse des risques liés à l'utilisation des services du cloud. Cette pratique permet donc :

- D'identifier les besoins d'affaires ;
- De s'assurer que le cloud constitue la meilleure solution pour le développement de l'application ;
- De s'assurer que le projet est aligné sur la vision de l'entreprise ;
- D'identifier les solutions et services de cloud potentiels.

Cette pratique est ajoutée afin de mieux cerner non seulement les avantages de l'utilisation du PaaS mais aussi les risques liés à son utilisation. Une mauvaise analyse de ces aspects peut occasionner des problèmes de verrouillage lors du développement des fonctionnalités.

Pratique 2 : Analyser des questions de sécurité et de confidentialité

Les questions de sécurité et de confidentialité sont importantes dans le choix du développement d'application dans le cloud. Cette pratique vise à s'assurer que l'application à réaliser ne manipule pas des données confidentielles et sensibles qui ne doivent pas être stockées chez de tiers fournisseurs de services du cloud. La validation du choix de développement d'application dans le cloud est faite à ce niveau.

Pratique 3 : Évaluer les plates-formes du cloud et choisir les fournisseurs

Cette pratique consiste à évaluer l'ensemble des plates-formes, services et modèles de déploiement nécessaires au développement de l'application. Elle connaît la participation de l'ensemble de l'équipe Scrum. En fonction des besoins en plate-forme, les fournisseurs de services sont évalués et sélectionnés. Comme activités dans cette pratique :

- Analyser les plates-formes PaaS pour le développement de l'application
- Identifier les outils de gestion de cycle de développement agile ALM (Application life cycle management) ;
- S'assurer que l'ensemble de l'équipe maîtrise les outils afin de garantir la transparence et l'inspection du processus Scrum ;
- Identifier une liste de fournisseurs de services répondant aux besoins ;
- Choisir les fournisseurs.

À ce niveau aussi, une mauvaise évaluation des besoins en plate-forme et des fournisseurs peut entrainer au des problèmes qui retarderont la livraison des fonctionnalités. Certains critères évoqués dans la section 5.5.2.1 permettront de faire une meilleure sélection des services afin

de minimiser les risques potentiels dus au manque de contrôle de l'infrastructure dans un environnement PaaS.

Pratique 4 : Gérer les contrats et acquérir les services

Cette pratique est importante pour réussir le développement d'application dans le cloud. Elle permet de s'assurer que l'équipe Scrum dispose de tous les environnements, services et outils pour livrer l'incrément. Les contrats sont négociés par les **Cloud broker** à travers les activités ci-après :

- Négocier le niveau de service et les prix ;
- Valider la confidentialité des données, la conformité et les pratiques d'audit ;

Acquérir les services.

Cette pratique dans le but de mieux gérer les services.

Pratique 5 : Définir les accès aux services et configuration des environnements

Ici, il s'agit de rendre accessibles les plates-formes et environnements aux membres d'équipe. Chaque membre de l'équipe Scrum a donc à sa disposition les ressources nécessaires pour livrer les incréments à chaque itération. Les activités suivantes sont effectuées.

- Configurer les accès aux services et environnements ;
- Configurer les environnements de développement ;
- Configurer les environnements de tests.
- Configurer l'environnement de production

Pratique 6 : Planifier les activités

Après avoir identifié les services et plates-formes nécessaires à la livraison de l'incrément, il faut élaborer le plan d'acquisition des services. Elle connaît la participation de tous les membres de l'équipe Scrum. Les activités suivantes sont menées :

- Planifier la gestion des contrats avec les fournisseurs ;
- Planifier la gestion des risques liés à l'utilisation des plates-formes du cloud;
- Planifier les activités d'acquisition des plates-formes.

Pratique 7 : Construire le code et composer les services

Comme évoque plus haut, le développement d'applications dans le PaaS revient à utiliser les outils et services et environnement du PaaS pour la création d'applications SaaS. Cette pratique implique les activités suivantes :

- Éditer le code source et le publier sur l'environnement PaaS;
- Éditer le code source directement dans le PaaS en utilisant les outils de développement du fournisseur ;
- Associer les services et API nécessaires à l'application ;
- Contrôler les configurations de l'application ;
- Générer ou construire les exécutables de l'application.

Pratique 8 : Effectuer les tests en parallèle

Les activités de développement et de test se font généralement de façon classique, mais les services du cloud offrent la possibilité d'effectuer les tests en parallèle afin d'accélérer le développement. Ceci est possible à cause de la facilité à créer les environnements dans le cloud. Pour cette pratique, les actions suivantes sont menées :

- Créer rapidement et facilement plusieurs environnements de tests ;
- Utiliser les services de tests standards (TaaS) fournis par le cloud;

- Effectuer les tests de charge et performance afin d'assurer la qualité de l'application ;
- Surveiller l'augmentation rapide de la consommation des ressources disponibles.

Pratique 9 : Automatiser les tests

Les plates-formes de développement du cloud offrent également des outils pour automatiser certaines activités de tests. Cette pratique permet de réduire les efforts, du temps et par conséquent augmenter la vélocité de l'équipe. Ainsi, les développeurs et testeurs effectuent les actions suivantes :

- Planifier les tests ;
- Élaborer les cas de tests ;
- Exécuter les tests.

Pratique 10 : Automatiser les rétroactions

Il est important de bien gérer les rétroactions de l'équipe, des parties prenantes et des utilisateurs finaux. Cette pratique consiste donc à utiliser les outils et ressources du cloud pour les automatiser. Ces données de rétroactions permettent aux développeurs d'améliorer le processus de développement dans le cloud. Ces outils encore appelés « **Feedback Driven Development** » sont intégrés aux environnements de développement par les fournisseurs. Les activités suivantes sont effectuées :

- Configurer les outils d'automatisation des rétroactions ;
- Collecter les données de rétroactions ;
- Tenir compte de ces rétroactions lors des prochaines itérations.

Pratique 11 : Auditer les services de cloud

Les activités de contrôle et de suivi des services sont effectuées durant chaque itération **« Cloud Auditor ».** Cette pratique s'effectue de façon continue durant pour la capture et la surveillance des données**.** Ceci leur permet de mesurer et évaluer les différents services en cours d'utilisation afin de s'assurer que les fournisseurs respectent le niveau de service requis. Les activités suivantes sont effectuées :

- Mesurer et évaluer les services de façon continue ;
- Élaborer des rapports d'audit sur les plates-formes ;
- Rendre accessibles les rapports d'audits aux membres d'équipe Scrum.

Pratique 12 : Analyser les besoins de mise à l'échelle

Avec la volatilité des demandes de changements, les besoins en ressources et en environnements peuvent changer d'une itération à une autre. Ainsi, l'équipe de développement peut avoir besoin de ressources complémentaires. Cette pratique est assurée par les **Cloud auditor** et **Cloud broker** et permet donc :

- De recueillir et analyser les demandes en plate-forme des développeurs et testeurs ;
- D'identifier les services à mettre à l'échelle.

Pratique 13 : Déployer une version de l'application

Le but ultime du développement d'application dans le cloud est de rendre disponibles rapidement les applications aux utilisateurs en réduisant le temps de mise en marché. Ainsi, lorsqu'une version de l'application est validée, elle est livrée puis publiée dans le cloud pour être accessible aux utilisateurs. Cette pratique concerne l'ensemble des parties prenantes du projet et les activités suivantes sont effectuées :

- Valider l'application ;
- Configurer les environnements de production ;

- Rendre l'application disponible dans le cloud;
- Effectuer les tests de disponibilité.

Pratique 14 : Traiter les rétroactions des utilisateurs

Cette pratique permet de collecter les rétroactions des utilisateurs afin d'en tenir compte pour les prochains itérations et incréments. En effet le développement d'application facilite le processus d'intégration continue. Elle rejoint la pratique d'automatisation des rétroactions qui utilisent les ressources du cloud.

Pratique 15 : Mettre à l'échelle l'environnement de production

Cette pratique est habituellement assurée par les équipes des opérations. Ils travaillent en collaboration avec l'équipe de développement Scrum, en exécutant les tâches suivantes :

- Analyser les besoins de mise à l'échelle des ressources de l'application ;
- Souscrire au service du cloud;
- Planifier les activités de mise à l'échelle des ressources ;
- Configurer les nouvelles ressources ;
- Effectuer les tests de disponibilité ;
- Rendre accessible l'application ;
- Collecter et traiter les rétroactions des utilisateurs.

Synthèse

L'ensemble des pratiques proposées dans notre méthode viennent répondre au besoin de maîtriser les spécificités de l'environnement du PaaS. L'infrastructure n'étant plus gérée par l'équipe du projet de développement, il important de bien analyser les contraintes et impacts de cet environnement sur le succès du projet. L'utilisation de l'environnement PaaS offre

l'avantage de permettre à l'équipe du projet de se concentrer sur les activités de gestion et de développement des fonctionnalités, tout en optimisant les coûts liés aux infrastructures. Néanmoins il important comme évoqué ci-haut de planifier et de maitriser quelques risques liés à cet environnement, notamment la dépendance vis-à-vis des fournisseurs de services qui contrôlent entièrement l'infrastructure.

5.6 Conclusion du chapitre

Dans ce chapitre, la méthode Scrum a été sélectionnée pour l'applicabilité du développement agile dans le cloud. Dans un premier temps, la méthode Scrum Cloud basée sur la méthode Scrum du SBOK a été présentée. Un certain nombre de modifications ont été identifiées afin répondre aux exigences du cloud. Ainsi, six évènements et 15 pratiques supplémentaires ont été identifiés. L'implication du PaaS a été présentée dans chaque évènement afin de montrer le processus de développement.

Parmi ces évènements, celui de souscription de services du cloud a été ajouté pour répondre aux besoins en environnements, outils et ressources. Aussi, de nouveaux rôles tels que « **Cloud broker** » et « **Cloud auditor** » ont été ajoutés dans le but de permettre à l'équipe Scrum de livrer plus facilement l'incrément. Ces rôles sont importants au démarrage du projet pour s'assurer que le cloud répond aux besoins d'affaires.

CHAPITRE 6

ANALYSE DE L'IMPCAT DU PAAS SUR LES OUTILS DE GESTION SCRUM

6.1 Introduction

Ce chapitre aborde les effets de l'utilisation de l'environnement PaaS sur les outils de gestion de Scrum. Quelques outils clés de la méthode Scum sont analysés. La gestion de risque est aussi absorbée afin d'identifier et évaluer les risques potentiels engendrés par l'utilisation du PaaS.

6.2 Impact du PaaS sur les outils de gestion de Scrum

Dans l'application de la méthode Scrum, des artéfacts et outils sont utilisés par les équipes pour livrer les fonctionnalités. Cette section présente l'impact de l'utilisation de l'environnement PaaS sur ces différents éléments :

Tableau 6.1 Outils de gestion Scrum

	Outils de gestion
1.	Backlog de produit
2.	Backlog de sprint
3.	Gestion de l'avancement
4.	Gestion des changements
5.	Gestion des risques
6.	Gestion des approvisionnements

6.2.1 Backolg de produit

Le Backlog de produit représente la liste dynamique des exigences, fonctions et améliorations nécessaires à la réalisation de l'application. Ces différents items sont continuellement actualisés et ordonnés.

L'utilisation de l'environnement de PaaS n'implique pas de changements au niveau du contenu du Backlog et de son fonctionnement. Néanmoins la plupart des fournisseurs de services PaaS offrent des applications SaaS qui permettent de gérer le Backlog de produit et surtout de faire le lien avec l'environnement PaaS de développement pour faciliter le suivi en temps réel des activités.

6.2.2 Backolg de Sprint

Comme le Backlog de produit, le Backlog de Sprint retrace l'ensemble des items nécessaires pour livrer l'incrément. Cette liste est plus détaillée que celle du Backlog de produit. Les applications SaaS sont aussi proposées pour gérer avec transparence cet outil.

6.2.3 Gestion de l'avancement

Pour suivre la progression du projet, plusieurs outils de projections de tendances comme le « **Burn Down Chart** » et le « **Burn Up Chart** » sont utilisés. Ces 2 outils sont utilisés pour mesurer l'évolution de chaque itération et de l'ensemble de l'incrément (« **Realease »**). Plusieurs formes et types de diagrammes sont exploités pour représenter ces outils. D'autres techniques comme celui de la « **Valeur acquise** » sont aussi utilisées. Cette dernière mesure l'avancement dans le temps en fonction des prévisions et des efforts réels effectués.

La figure 6.1 nous montre un exemple de « **Burn Down chart** » pour une itération. Elle présente la somme des efforts restants à effectuer et la progression idéale des efforts. Ces derniers sont évalués en heures (parfois en points). Sur cette figure on observe qu'à la date du 2017-09-18, l'itération est en retard, car l'effort restant dépasse l'idéal.

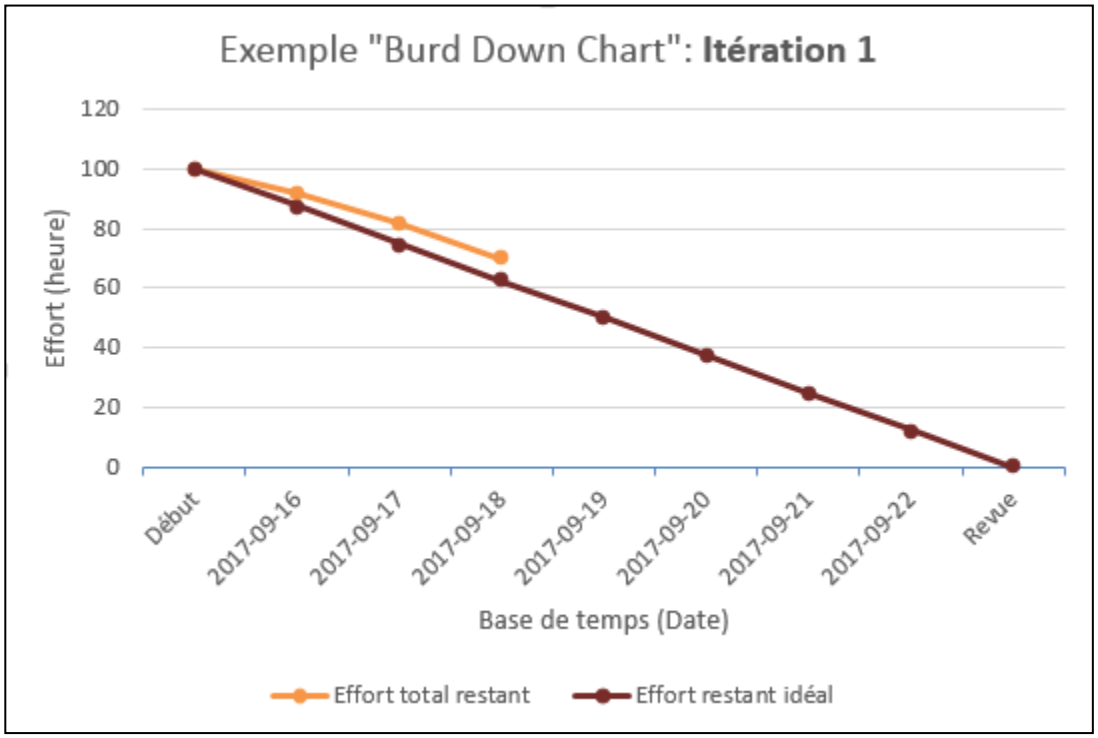

Figure 6.1 Exemple de « **Burn Down Chart** » pour une itération

Le « **Burn Down Chart** » est souvent le plus utilisé et peut être représenté sous plusieurs formes grâce aux applications (SaaS) proposées par les fournisseurs. Ces outils facilitent la représentation et l'interprétation de ces graphes. L'utilisation de l'environnement du PaaS n'a pas d'influence directe sur ces outils, mais la mise à disposition de ces applications SaaS par les fournisseurs constitue un avantage.

6.2.4 Gestion des changements

L'un des principes clés des méthodes agiles et en occurrence de Scrum est l'acceptation des demandes de changement en tout temps lors de la réalisation de l'application. Ces demandes peuvent intervenir pendant tous les évènements présentés dans notre méthode. Les demandes de changements sont souvent faites par les parties prenantes pour répondre à la volatilité du marché, la concurrence ou d'autres facteurs déterminants pour le succès du projet. Parfois ces projets font partie de programme ou de portefeuille qui ont des contraintes particulières. Ces

demandes peuvent être mineures ou majeures (influençant l'issu du projet). Les demandes mineures peuvent être approuvées par le **Product Owner,** par contre celles qui sont majeures nécessitent l'intervention des parties prenantes (commanditaire, client, etc.). Pour une bonne gestion de ces demandes, il est important d'adopter des processus formels de gestion.

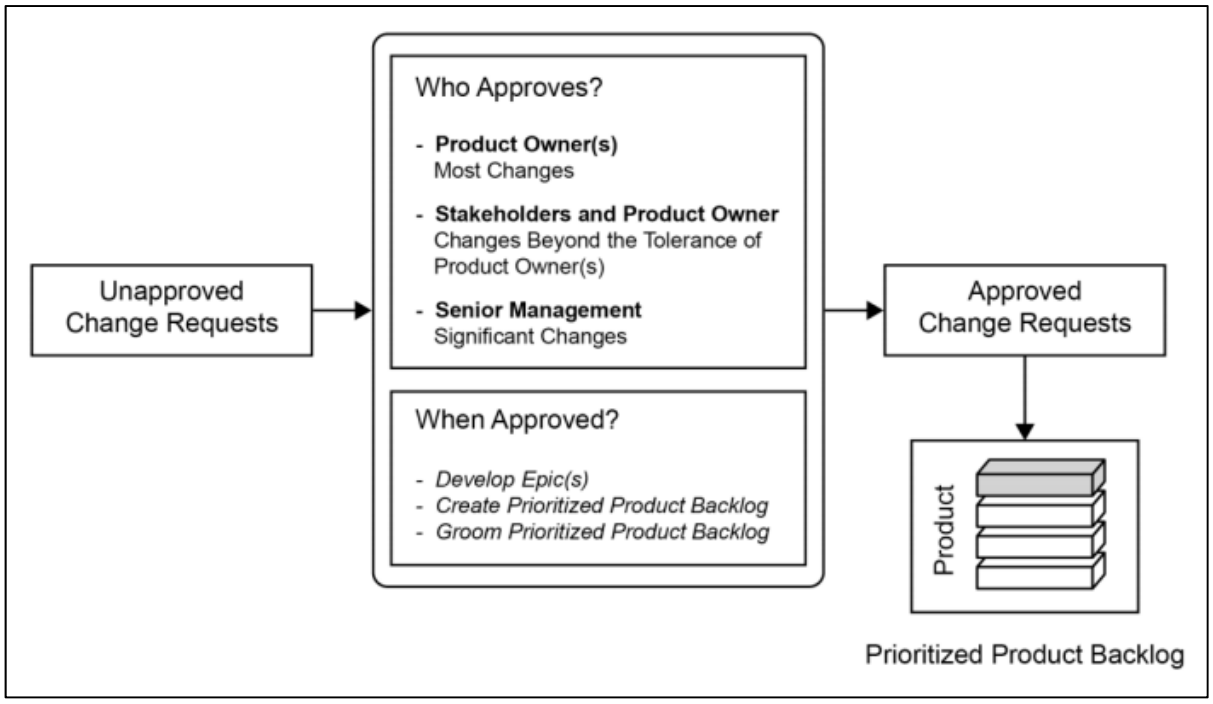

Figure 6.2 Exemple de processus d'approbation des demandes de changement
Tirée de (SCRUMstudy, 2016)

Les demandes de changement après leur approbation sont intégrées dans le carnet de produit afin d'être prises en compte dans les prochaines itérations (par priorité).

Le fait de choisir de réaliser le projet agile dans un environnement PaaS, implique déjà quelques contraintes liées à l'infrastructure. Rappelons que dans un environnement PaaS, l'équipe de projet le pipeline de développement (serveurs, outils de développement, de test, langages) est géré par le fournisseur. Des demandes de changement majeures qui impactent ces éléments du pipeline de développement sont très difficiles à être validées. Elles peuvent nécessiter des changements de fournisseurs ou de type de services cloud. À cet effet, il est très

important comme nous l'avons évoqué plus haut de tenir compte d'un certain nombre de paramètres clés pour le choix des fournisseurs. Les problèmes de **verrouillage** sont observables dans ces cas de figure.

6.2.5 Gestion des risques

Selon le guide du SBOK, Scrum est particulièrement utile pour les projets ayant beaucoup d'incertitudes (SCRUMstudy, 2016). Ces incertitudes entrainent donc des risques potentiels à gérer. Le risque se définit comme un évènement incertain qui peut affecter l'atteinte des objectifs du projet. Ces risques sont caractérisés par une probabilité d'apparition et l'impact potentiel sur le projet. La gestion des risques dans un projet impliquent les 5 étapes suivantes (SCRUMstudy, 2016) :

1. Identification des risques ;
2. Évaluation et estimation des risques identifiés ;
3. Priorisation des risques ;
4. Atténuation des risques ;
5. Communication sur les risques.

Identification des risques

Elle consiste à déterminer qualitativement les évènements capables d'impacter le projet. Elle se fait non seulement au début du projet, mais aussi lors de chaque itération. Plusieurs outils sont utilisés pour l'identification des risques et sont implémentés dans la plupart des applications proposées par les fournisseurs de services. Le « **Risk Breakdown Structure** » ou structure de répartition des risques est l'un des outils utilisés. Dans la section 5.5.2, nous avons identifié quelques risques potentiels engendrés par l'utilisation de l'environnement PaaS. Ces risques sont à prendre en compte déjà au début du projet de développement :

Tableau 6.2 Liste des risques potentiels liés au PaaS

	Risques	**Causes potentielles**
Risque 1	Risque de verrouillage par les fournisseurs PaaS	En quête de client, les fournisseurs peuvent créer une dépendance de leurs services en ne facilitant pas la migration des données (utilisation de services non standards)
Risque 2	Risque de disponibilité des services	Défaillances au niveau des fournisseurs
Risque 3	Risque de sécurité des données	Manque de maîtrise des procédures de sécurisation du pipeline de développement proposé par le fournisseur
Risque 4	Risque d'augmentation des coûts des services	L'utilisation de services propriétaires peut entrainer une augmentation des coûts de ses services (usage de services propriétaires nécessitant des licences)
Risque 5	Risque de changement de pipeline de développement	Des demandes de changement qui ont impact majeur sur le pipeline de développement contrôlé par le fournisseur

Évaluation et estimation des risques

Cette activité permet d'évaluer la probabilité d'apparition des risques et leurs impacts éventuels sur le projet. Si l'impact d'un risque est majeur, il peut entraîner l'arrêt ou la modification des objectifs du projet. Plusieurs techniques peuvent être utilisées pour l'évaluation des risques (rencontres, arbre de probabilité, diagramme de Pareto, matrice de probabilité et impact).

Pour les risques identifiés ci-haut dans un contexte d'utilisation de l'environnement PaaS, leurs probabilités et impacts dépendent de plusieurs facteurs : les contraintes du projet, la maturité et la qualité des services des fournisseurs.

Priorisation des risques

Après l'identification et l'évaluation, les risques sont ordonnés par ordre de priorité. Cette activité gérée par le « **Product Owner** », s'effectue lors des activités de priorisation du carnet de produit. Voici un exemple illustratif du processus de priorisation des risques :

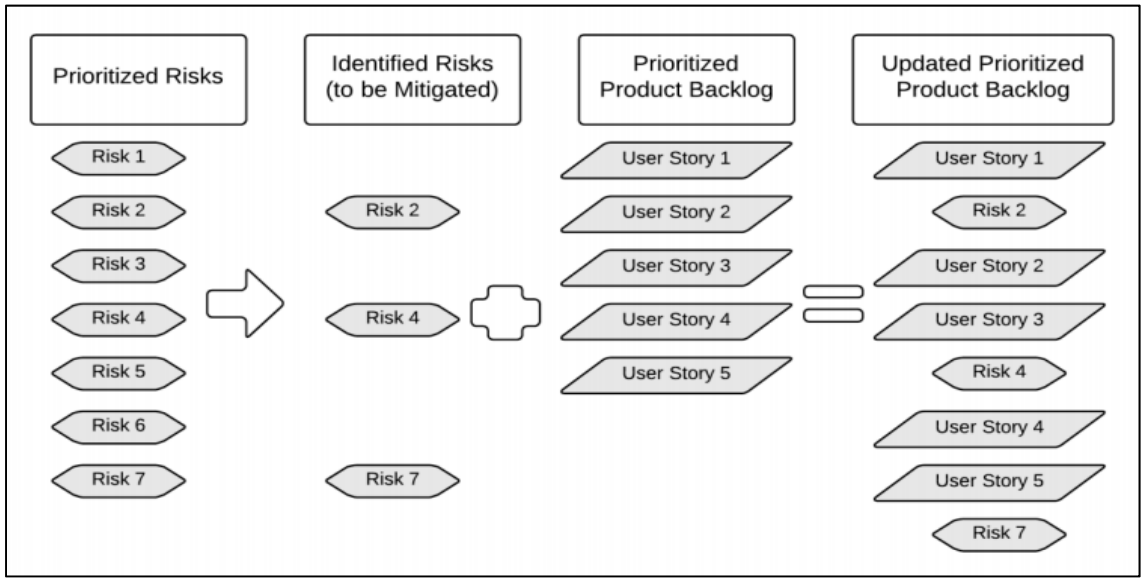

Figure 6.3 Processus de priorisation des risques
Tirée de (SCRUMstudy, 2016)

La figure ci-dessous nous montre un ensemble de risques ordonnés par ordre de priorité, dont 3 ont été choisis pour être atténués et insérés dans le carnet de produit en vue d'être pris en compte dans les prochaines itérations. Les risques liés à l'utilisation de l'environnent du PaaS que nous avons identifié plus haut seront priorisés et traités suivant ce processus afin d'atténuer leurs impacts.

Atténuation des risques

Elle consiste à déterminer les réponses à chaque risque en fonction de sa probabilité et son impact. Des plans B sont identifiés afin d'atténuer les impacts de ces risques sur le projet. Par défaut, l'utilisation de la méthode Scrum contribue déjà à l'atténuation des risques, car elle fait usage de cycle de développement court qui facilite leur détection rapide. Cette activité est menée par le « **Product Owner** » qui en tient compte pour la planification des prochaines itérations. Dans le cas des risques identifiés, nous proposons les approches de solutions suivantes :

Tableau 6.3 Liste des solutions aux risques PaaS

	Risques	**Approches de solutions**
Risque 1	Risque de verrouillage par les fournisseurs PaaS	1. Bonne maîtrise de l'architecture du PaaS par l'équipe du projet 2. Bonne sélection des fournisseurs 3. Utilisation de services cloud standards et non-propriétaires pour réduire le risque de verrouillage 4. Avoir un plan de migration des données 5. Tenir compte des rôles de « cloud broker » et « cloud auditor » dans la composition de l'équipe du projet 6. Contrôler en permanence le niveau de qualité des services conformément au **SLA**
Risque 2	Risque de disponibilité des services	
Risque 3	Risque de sécurité des données	
Risque 4	Risque d'augmentation des coûts des services	
Risque 5	Risque de changement de pipeline de développement	

Communication sur les risques

La communication sur la rentrée est la dernière étape de la gestion des risques. Elle s'effectue tout au long du cycle de développement et permet d'assurer la transparence du projet et d'impliquer davantage l'ensemble des parties prenantes. Les risques peuvent être aussi abordés par l'équipe de développement lors des rencontres quotidiennes. Le principal outil utilisé pour la communication sur les risques est le « **Risk Burndown Chart** ». La figure ci-dessous nous montre l'exemple d'un graphe d'évolution des risques en fonction des itérations. Elle présente l'évolution des probabilités cumulées de 6 risques jusqu'à la cinquième itération. Elle montre aussi une diminution progressive de la sévérité des risques.

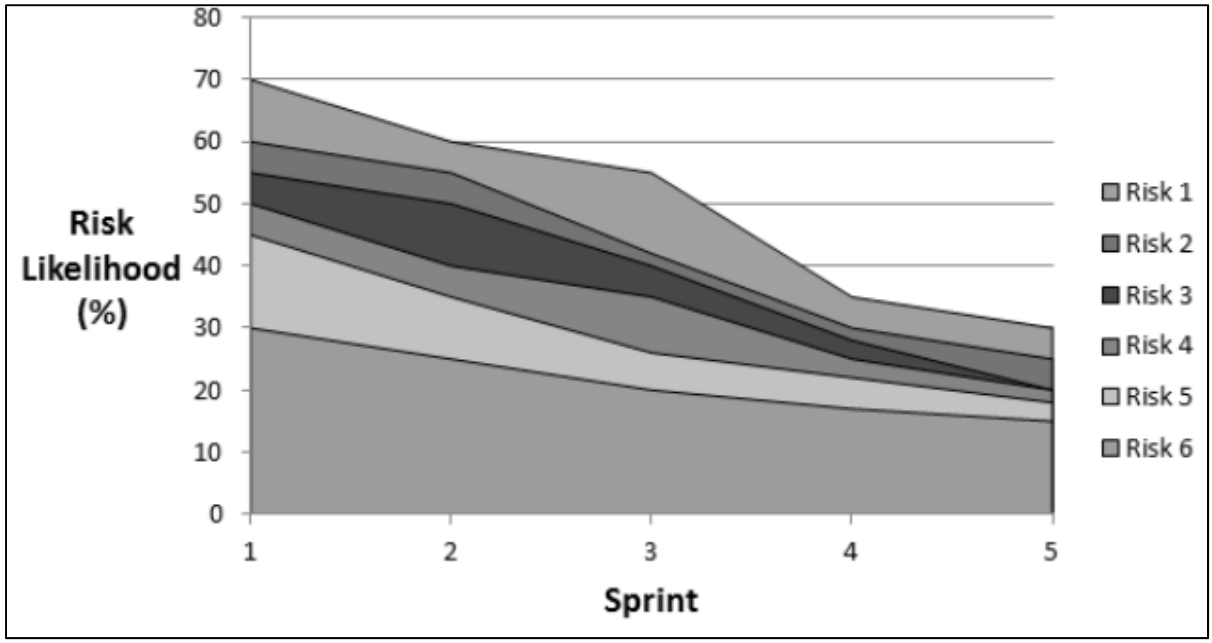

Figure 6.4 Exemple de diagramme de risque « **Risk Burndown Chart** »
Tirée de (SCRUMstudy, 2016)

Cet outil est souvent disponible dans la plupart des applications de gestion de projets Scrum ou Agile en mode SaaS.

6.2.6 Gestion des approvisionnements

Lors de l'exécution des projets, certains approvisionnements sont faits pour assurer la livraison des produits. Dans le cas du développement dans un environnement PaaS, les approvisionnements sont essentiellement liés à l'acquisition des services cloud et à la gestion des contrats.

Un bon plan de gestion des contrats notamment les **Cloud Service Agrement (CSA)** afin gérer au mieux ces services cloud et faciliter leur évaluation.

6.3 Conclusion du chapitre

Dans ce chapitre, nous avons analysé l'impact de l'utilisation de l'environnement du PaaS sur quelques outils de Scrum comme le Backlog de Produit et de Sprint, ainsi que d'autres outils de gestion. Nous avons donc identifié quelques risques induits par l'utilisation de l'environnement PaaS. Quelques outils de gestion des risques ont été proposés. Dans la suite du document, il est question de présenter les contributions de notre recherche ainsi que les limites et les recommandations pour de futurs travaux similaires.

CHAPITRE 7

CONTRIBUTIONS DE LA RECHERHCE

7.1 Retour sur la méthode proposée

Dans la première partie de cette recherche nous avons approfondi nos connaissances sur le développement d'applications agile dans le cloud afin de voir son adoption par les entreprises.

Dans un second temps, conformément à nos objectifs, nous avons donc proposé une méthode basée sur la méthode Scrum. Cette méthode est étendue aux spécificités de l'environnement du PaaS (section 5.5.1). Nous nous sommes basés sur le guide Scrum SBOK (2016) qui définit l'ensemble des phases, pratiques et rôles nécessaires à l'utilisation de la méthode Scrum. Nous avons associé les contraintes et avantages de l'environnement du PaaS aux pratiques de Scrum afin de déterminer les modifications à apporter pour démontrer l'applicabilité de développement d'applications agile dans le PaaS.

Le guide du SBOK préconise cinq phases, tandis que nous dans notre méthode, nous avons ajouté une phase ou évènement de « **pré-projet** » qui est primordial dans la prise en compte des risques et contraintes liés au PaaS. L'évènement initial de « **pré-projet** » permet de s'assurer que le PaaS répond aux besoins d'affaires du projet de développement logiciel. Cet évènement connaît la participation de l'ensemble de l'organisation pour analyser les risques liés à l'utilisation du PaaS. Nous estimons que cette étape est crucial pour toutes les entreprises qui veulent migrer le développement de leur application dans le cloud. Nous avons donc identifié quelques risques dont le manque de contrôle de l'infrastructure par les équipes de développement. Les autres évènements ont connu l'ajout d'autres pratiques supplémentaires.

Le second évènement « **démarrage et souscription cloud** » a été étendu afin gérer les activités d'acquisition et de gestion des services du cloud. À ce niveau nous avons identifié l'ensemble des facteurs à prendre en compte pour la sélection des fournisseurs et des services du cloud.

Les autres évènements « **développement** », « **revue et rétrospection** » et « **livraison** » demeurent quasiment identiques à ce qui est proposé dans le guide du SBOK. Néanmoins, nous avons mis en exergue quelques pratiques particulières à appliquer pour faciliter le développement dans un environnement PaaS.

Dans notre méthode, nous avons aussi ajouté deux rôles complémentaires qui doivent être pris en compte dans les équipes de développement. Il s'agit des rôles de « **cloud auditor** » et « **cloud broker** » qui sont définis dans la norme ISO 17789:2014 et qui sont relatifs à l'architecture du cloud computing. Nous avons introduit ces 2 deux rôles afin de faciliter la compréhension, l'acquisition et la gestion des services PaaS. L'ajout de ces rôles rejoint en partie la proposition de Patidar (2011) qui proposait une méthode agile étendue qui intègre le rôle de « **cloud provider** » aux différentes étapes clés (planification, conception, développement et test) du cycle de développement agile (section 2.5). Notre travail vient donc apporter plus de précision sur les rôles du cloud à intégrer.

Cette méthode proposée vient répondre aux objectifs de notre travail de recherche. Les activités menées ont contribué à faire évoluer la thématique de développement agile dans le cloud et en particulier dans un environnement PaaS. Ce dernier qui permet de réduire les coûts liés aux infrastructures et qui permet aux équipes de projet de se focaliser d'avantages sur le développement et la livraison rapide des applications.

7.2 Nos contributions

- **Une méthode qui étend la méthode Scrum**

Notre objectif dans cette recherche étant de vérifier l'applicabilité des méthodes agiles dans un environnement cloud, nous avons proposé une extension de la méthode Scrum dans un environnement PaaS. De plus en plus, les entreprises choisissent de migrer leurs activités de développement dans le cloud. Notre travail vient donc compléter un ensemble de travaux similaires proposés par d'autres chercheurs. Ce travail vient confirmer les nombreux avantages

de l'utilisation conjointe de l'approche agile et du cloud pour le développement logiciel. Ces avantages sont évoqués dans les travaux de S. Kalem (2013) qui ont combiné la méthode DSDM et l'environnement PaaS (Google App Engine) :

« The main benefits when using agile methods in conjunction with cloud computing include: Increased quality of application, Effective use of resources, Lowered time-to-market, Cost savings […]. The agile development is compared with agile development in conjunction with cloud computing. Comparison is made upon two types of development. The one is using only agile methods and the second one is using agile methods and cloud computing. » (Kalem, Donko, & Boskovic, 2013b).

Contrairement au travail de S. Kalem (2013), notre méthode aborde les différents aspects (évènements, rôles et pratiques) de gestion à modifier pour tirer profit de ces avantages.

Notre travail est quasiment similaire aux travaux récents de R. Singhal (2016) qui a également travaillé sur l'intégration du cloud dans chacune des phases du guide du SBOK. Néanmoins sa méthode n'est pas spécifique à l'environnement PaaS et il n'a pas proposé l'évènement initial « pré-projet » pour l'analyse des besoins en cloud. Sa méthode ne fait pas aussi le détail des rôles au sein des équipes. Nous estimons que ces aspects sont importants et viennent en continuité au travail de R. Singhal (2016).

- **Méthode qui met en lumière les risques d'utilisation du PaaS**

Notre méthode étant basée sur l'environnement du PaaS, elle contribue donc à la réduction des coûts liés à l'infrastructure et permet aux équipes de développement de consacrer le maximum du temps sur les activités de développement et de livraison des fonctionnalités. Rappelons que dans un environnement PaaS, toute l'infrastructure est à la charge du fournisseur.

Malgré cet avantage que constitue la gestion de l'infrastructure par le fournisseur, notre recherche à contribuer à identifier certains risques que peuvent rencontrer les projets de

développement dans le PaaS. Ces risques n'ont pas été identifiés dans les travaux de R. Singhal (2016).

Nous espérons que notre méthode bien que théorique permettra aux équipes agiles de disposer des informations de base nécessaires à la gestion des projets de développement dans un environnement PaaS et facilitera la prise en compte des risques potentiels.

CHAPITRE 8

DISCUSSIONS

8.1 Limitation de notre méthode

Dans la section 5.2 nous avons présenté le choix de la méthode agile utilisée dans notre recherche. Le choix de la méthode Scrum au détriment des autres méthodes agiles populaires représente sans doute une limitation à notre travail de recherche. Ceci ne permet donc pas de conclure sur l'applicabilité de toutes les méthodes agiles de développement dans le cloud.

La méthode Scrum étant une méthode assez légère qui définit juste le minimum d'activités, de rôles et d'artéfacts, nous ne pouvons pas généraliser notre solution aux autres méthodes agiles sans d'autres études supplémentaires.

Dans notre recherche nous n'avons pas travaillé sur les aspects techniques du développement d'applications dans le cloud. D'autres approches comme DevOps (chapitre 3) traitent d'avantages des aspects techniques et des outils utilisés pour le développement d'applications dans le cloud. Il faut noter aussi l'approche DevOps propose en plus du développement dans le PaaS, le développement dans un environnement IaaS ou hybride qui entrainent d'autres avantages et contraintes qui sont à explorer.

Malgré les avantages que peut apporter notre méthode pour le développement d'applications dans un environnement PaaS pour les équipes, il existe quelques risques potentiels que nous avons identifiés dans notre recherche. Nous n'avons pas effectué une étude approfondie sur ces risques afin de contrôler leur impact sur le succès d'un projet de développement d'applications dans le cloud. Ces risques sont identifiés, mais nous n'avons pas procédé à leur quantification et contrôle en situation de projets.

Notre méthode n'a fait pas l'objet d'une utilisation entreprise. Ceci ne nous permet donc pas de faire des analyses complémentaires sur la base de données recueillies lors de l'exécution d'un projet réel de développement d'applications dans un environnement PaaS.

8.2 Continuation de la recherche

Comme évoqué au début de notre travail de recherche, notre objectif est d'étudier l'applicabilité des méthodes agiles dans le cloud. Malgré les résultats obtenus, nous avons identifié dans la section précédente quelques limites qui nous amènent donc à faire des suggestions pour la continuation de ce travail.

Notre méthode n'ayant pas fait l'objet d'une application en situation de projet, ce travail de recherche pourrait être poursuivi afin de valider les résultats théoriques obtenus. Une expérimentation de notre méthode est donc souhaitée.

Notre méthode est basée sur la méthode Scrum, et il serait intéressant que d'autres travaux de recherches similaires se consacrent à d'autres méthodes agiles comme XP, DSDM et FDD. Ceci permettra d'avoir une base commune pour comparer ces méthodes dans un contexte de PaaS. Les critères d'analyse et de comparaison se baseront sur les contraintes et avantages du cloud.

Dans notre recherche nous n'avons pas abordé les aspects techniques du développement dans le cloud. D'autres approches de développement dans le cloud comme DevOps sont à étudier en profondeur pour vérifier l'applicabilité du développement dans les différents types d'environnements du cloud (PaaS, IaaS, SaaS ou hybride),

Pour finir, nous avons identifié quelques risques majeurs liés à l'utilisation de l'environnement PaaS. En effet, le manque de contrôle de l'infrastructure par les équipes de développement est l'un des risques qui peut occasionner des problèmes de verrouillage des fournisseurs. Nous suggérons donc que d'autres recherches similaires soient faites sur ces risques liés à

l’utilisation du PaaS afin d’atténuer leur impact sur le succès des projets de développement logiciel.

CONCLUSION

Les méthodes agiles sont devenues les plus utilisées dans le développement logiciel, car elles favorisent la livraison continue de logiciels fonctionnels répondant aux besoins des clients. Malgré ses avantages, elles connaissent beaucoup de challenges et de limitations. Pour y remédier, plusieurs entreprises de développement choisissent de migrer leurs activités dans le cloud qui connaît ces dernières années une croissance rapide. Cependant, pour bénéficier des avantages de l'utilisation des ressources du cloud, il est nécessaire d'évaluer l'applicabilité de ces méthodes agiles dans le cloud.

La revue de littérature a permis de montrer les bénéfices du développement agile d'applications dans le cloud. Pour tirer profit de ses avantages, nous avons proposé une méthode Scrum basée sur celui du SBOK. Cette méthode proposée est adaptée au développement agile d'applications dans l'environnement du PaaS. Elle redéfinit les différents évènements de la méthode Scrum en montrant l'implication des services du PaaS dans le développement d'applications. L'un des évènements clés ajoutés est celui de « **pré-projet** » qui décrit les activités et rôles nécessaires pour le choix du développement d'applications dans le PaaS. Nous avons également modifié l'évènement « démarrage » afin de faciliter une meilleure sélection, acquisition et la mise à jour des services du PaaS.

Malgré les résultats obtenus, il faut noter que pour ce travail de recherche, nous nous sommes limités au développement d'applications dans le PaaS. Ces limites ont été abordées en détail dans le dernier chapitre. Il faut noter également que l'approche DevOps est l'une des approches les plus utilisées actuellement dans le cloud pour le développement et l'exploitation des applications. Il est donc important que d'autres travaux de recherche se penchent sur le développement agile dans l'ensemble des 3 couches du cloud afin d'établir une grille de choix des services des services du cloud nécessaires pour le développement d'applications agile en fonction des besoins et de leur complexité.

L’autre problème issu de l’utilisation du cloud pour le développement agile de logiciels est le verrouillage par les fournisseurs de service. La migration du processus de développement agile dans le cloud crée une dépendance vis-à-vis du fournisseur de services et rend parfois difficile la portabilité des applications. Nous suggérons donc que d’autres recherches soient faites sur la standardisation des environnements, outils, services et processus de développement d’applications dans le PaaS. Pour finir, il est important aussi d’expérimenter cette méthode Scrum en situation réelle de projet de développement afin d’identifier ses potentielles limites et l’améliorer.

BIBLIOGRAPHIE

Abhishek, Reena, J., & Rani. (2012). Analytical Study of Agile Methodology with Cloud Computing. *IJCA Proceedings on National Workshop-Cum-Conference on Recent Trends in Mathematics and Computing 2011, RTMC*, 14.

Akerele, O., Ramachandran, M., & Dixon, M. (2013). Testing in the Cloud: Strategies, Risks and Benefits. Dans Z. Mahmood & S. Saeed (Éds.), *Software Engineering Frameworks for the Cloud Computing Paradigm* (pp. 165-185). London: Springer London. doi: 10.1007/978-1-4471-5031-2_8. Repéré à http://dx.doi.org/10.1007/978-1-4471-5031-2_8

Aubry, C. (2014). *Scrum : Le guide pratique la Méthode Agile la plus populaire*. Paris: Dunop.

Chana, I., & Chawla, P. (2013). Testing Perspectives for Cloud-Based Applications. Dans Z. Mahmood & S. Saeed (Éds.), *Software Engineering Frameworks for the Cloud Computing Paradigm* (pp. 145-164). London: Springer London. doi: 10.1007/978-1-4471-5031-2_7. Repéré à http://dx.doi.org/10.1007/978-1-4471-5031-2_7

Chung, Y., & Shao-Zong, C. (2013). DDMan: a management system for distributed software development in cloud computing environments. *International Journal of e-Education, e-Business, e-Management and e-Learning, 3*(6), 446-450. doi: 10.7763/IJEEEE.2013.V3.276. Repéré à http://dx.doi.org/10.7763/IJEEEE.2013.V3.276

Cocco, L., Mannaro, K., & Concas, G. (2012). A Model for Global Software Development with Cloud Platforms. Dans *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on* (pp. 446-452). doi: 10.1109/SEAA.2012.67

Coyne, S. S. B. (2015). DevOps pour les Nuls.

Cruzes, D. S., Moe, N. B., & Dybå, T. (2016). Communication between Developers and Testers in Distributed Continuous Agile Testing. Dans *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)* (pp. 59-68). doi: 10.1109/ICGSE.2016.27

DSDM Consortium. (2014). The DSDM Agile Project Framework v2. Repéré à https://www.dsdm.org/sites/default/files/The%20DSDM%20Agile%20Project%20Framework%20v2.pdf

Etchevers, X. (2012). *Deploying legacy applications in cloud computing environments* (Université de Grenoble). Repéré à Star Cnrs Univ-grenoble1 Lig Inpg Uga Univ-pmf_grenoble. (2012GRENM100). Repéré à https://tel.archives-ouvertes.fr/tel-00875568

Hamidi, K. (2014). *Agile software development, and its limitations* (M.S., California State University, Fullerton, Ann Arbor). Repéré à ProQuest Dissertations & Theses Global.

Hashmi, S. I., Clerc, V., Razavian, M., Manteli, C., Tamburri, D. A., Lago, P., . . . Richardson, I. (2011). Using the Cloud to Facilitate Global Software Development Challenges. Dans *2011 IEEE Sixth International Conference on Global Software Engineering Workshop* (pp. 70-77). doi: 10.1109/ICGSE-W.2011.19

Hibbs, C., Jewett, S., & William, M. (2010). *L'art du Lean : Software development*. Paris: Dunop.

ISO/IEC. (2014a). *Information technology — Cloud computing — Overview and vocabulary*.

ISO/IEC. (2014b). *Information technology — Cloud computing — Reference architecture*. 17789.

Jain, N., & Dubey, S. (2014). Agile Development Methodology with cloud computing. *International Journal Of Engineering And Computer Science, 3*(4), 6.

Kalem, S., Donko, D., & Boskovic, D. (2013a). Agile methods for cloud computing. Dans *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on* (pp. 1079-1083).

Kalem, S., Donko, D., & Boskovic, D. (2013b). Agile methods for cloud computing. Dans *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1079-1083).

Khan, P. M., & Beg, M. M. S. S. (2013). Extended Decision Support Matrix for Selection of SDLC-Models on Traditional and Agile Software Development Projects. Dans *Advanced Computing and Communication Technologies (ACCT), 2013 Third International Conference on* (pp. 8-15). doi: 10.1109/ACCT.2013.12

Mani, P., Deebitha.S, Jayakumar, S., & Gopalakrishnan, R. (2014). Enhancing Agile Software Development Using Cloud Computing: A Case Study. *International Journal of Research in Management & Business Studies, 1*(1), 3.

Mehrotra, S. (2015, 2015-12-01). Testing Agility in the Cloud: The 4Cs Framework. *LogiGEAR, IX*(4), 32.

Mwansa, G., & Mnkandla, E. (2014). Migrating Agile Development into the Cloud Computing Environment. Dans *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on* (pp. 818-825). doi: 10.1109/CLOUD.2014.113. Repéré à http://ieeexplore.ieee.org/ielx7/6968679/6973706/06973819.pdf?tp=&arnumber=6973819&isnumber=6973706

Paraiso, F. (2014). *socloud: distributed Multi-Cloud Platform for deploying, executing and managing distributed applications* (Université des Sciences et Technologie de Lille - Lille I). Repéré à Univ-lille3. Repéré à https://tel.archives-ouvertes.fr/tel-01009918

Patidar, S., Rane, D., & Jain, P. (2011). Challenges of software development on cloud platform. Dans *2011 World Congress on Information and Communication Technologies, WICT 2011, December 11, 2011 - December 14, 2011* (pp. 1009-1013). IEEE Computer Society. doi: 10.1109/WICT.2011.6141386. Repéré à http://dx.doi.org/10.1109/WICT.2011.6141386

Peter, M., & Timothy, G. (2011). The NIST Definition of Cloud Computing. *NIST Special Publication 800-145*, 7. Repéré à http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdfsite Web |URL| doi:DOI

Pramod, N., Muppalla, A. K., & Srinivasa, K. G. (2013). Limitations and Challenges in Cloud-Based Applications Development. Dans Z. Mahmood & S. Saeed (Éds.), *Software Engineering Frameworks for the Cloud Computing Paradigm* (pp. 55-75). London: Springer London. doi: 10.1007/978-1-4471-5031-2_3. Repéré à http://dx.doi.org/10.1007/978-1-4471-5031-2_3

Project Management Institute. (2013). *Project Management Body of Knowledge PMBOK Guide*. Newtown Square, Pennsylvania 19073-3299 USA: Project Management Institute.

Raj, G., Yadav, K., & Jaiswal, A. (2015). Emphasis on testing assimilation using cloud computing for improvised agile SCRUM framework. Dans *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)* (pp. 219-225). doi: 10.1109/ABLAZE.2015.7154995

Schwalbe, K. (2011). *Information technology Project management*. 20 Channel Center Street, Boston, USA: Course Technology.

SCRUMstudy. (2016). *A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK™ Guide)*. Repéré à www.scrumstudy.com

Singhal, R., Sonia, & Singhal, A. (2016). Rising future of Agile Software Development using Cloud Computing: A study using Cloud Computing in different phases of an Agile method-SCRUM. *International Journal of Computer Technology and Applications, 7*(2), 7.

Soojin, P., Mansoo, H., Sangeun, L., & Park, Y. B. (2015). A Generic Software Development Process Refined from Best Practices for Cloud Computing. *Sustainability, 7*(5), 5321-5344. doi: 10.3390/su7055321. Repéré à http://dx.doi.org/10.3390/su7055321

Tekool.net. (2016). Manifeste Agile. Repéré à http://www.tekool.net/blogfiles/manifeste-agile-imprimable/manifeste_agile_a3.pdf

The Standish Group International. (2013). *Chaos Manifesto*. The Standish Group International.

Tuli, A., Hasteer, N., Sharma, M., & Bansal, A. (2014). Empirical investigation of agile software development: a cloud perspective. *ACM SIGSOFT Software Engineering Notes, 39*(4), 6 pp. doi: 10.1145/2632434.2632447. Repéré à http://dx.doi.org/10.1145/2632434.2632447

Vaidya, D. D. K. (2011). An agile process framework for cloud application development. *CSC Papers*, 29.

VERSIONONE. (2015). *9TH ANNUAL State of Agile Survey*.

VERSIONONE. (2016). *The 10th annual State of Agile*.