Depuis maintenant quelques années (il est difficile de donner une date précise), les DSI s’appuient sur la mouvance agile afin de mener à bien leurs projets. Aujourd’hui, les patterns agiles arrivent à maturité et offrent un éventail de méthodologies adaptables à tous les contextes. Les méthodes agiles garantissent la satisfaction du client et non la conformité aux termes d’un contrat de développement. Elles sont centrées sur la satisfaction de besoin du client et non sur les termes contractuels du projet. Nous n’allons pas aborder en profondeur le concept de l’agilité, ceci n’est pas le propos de ce mémoire, mais nous allons tout de même faire un petit rappel des idées fortes de cette méthodologie. Il faut des **cycles courts**, quelques semaines tout au plus, et découper le projet en **petites tâches** puis les **hiérarchiser** **en fonction du besoin**. Cela permet d’éviter le superflu et de se concentrer au début de chaque cycle sur ce qui a de la valeur pour l’utilisateur final. Le **feedback** permanent devient la règle d’or, avec des validations à chaque étape et des techniques ludiques d’évaluation de l’utilité des fonctions. L’agilité offre une meilleure visibilité et permet d’éviter les dérives observées lorsque les développeurs sont isolés. Le **changement** est autorisé voir encouragé, même tardivement,  car c’est un avantage décisif pour le client. Cela permet de ne pas se priver des bonnes idées en cours de route et surtout d’éliminer les mauvaises idées lancées au début du projet. Les méthodes agiles favorisent la **co-construction**, en intégrant l’annonceur lui-même dans le travail quotidien et en responsabilisant la totalité de l’équipe de développement, créant ainsi un véritable **esprit collaboratif** et l’ensemble du projet en gagne en qualité.

Cependant l’agilité, lorsqu’elle est exclusivement cantonnée au développement, se trouve néanmoins freinée par les tâches d’exploitation. Le mouvement DevOps a pour objectif d’étendre les pratiques agilistes à la livraison et au déploiement du projet.

DevOps

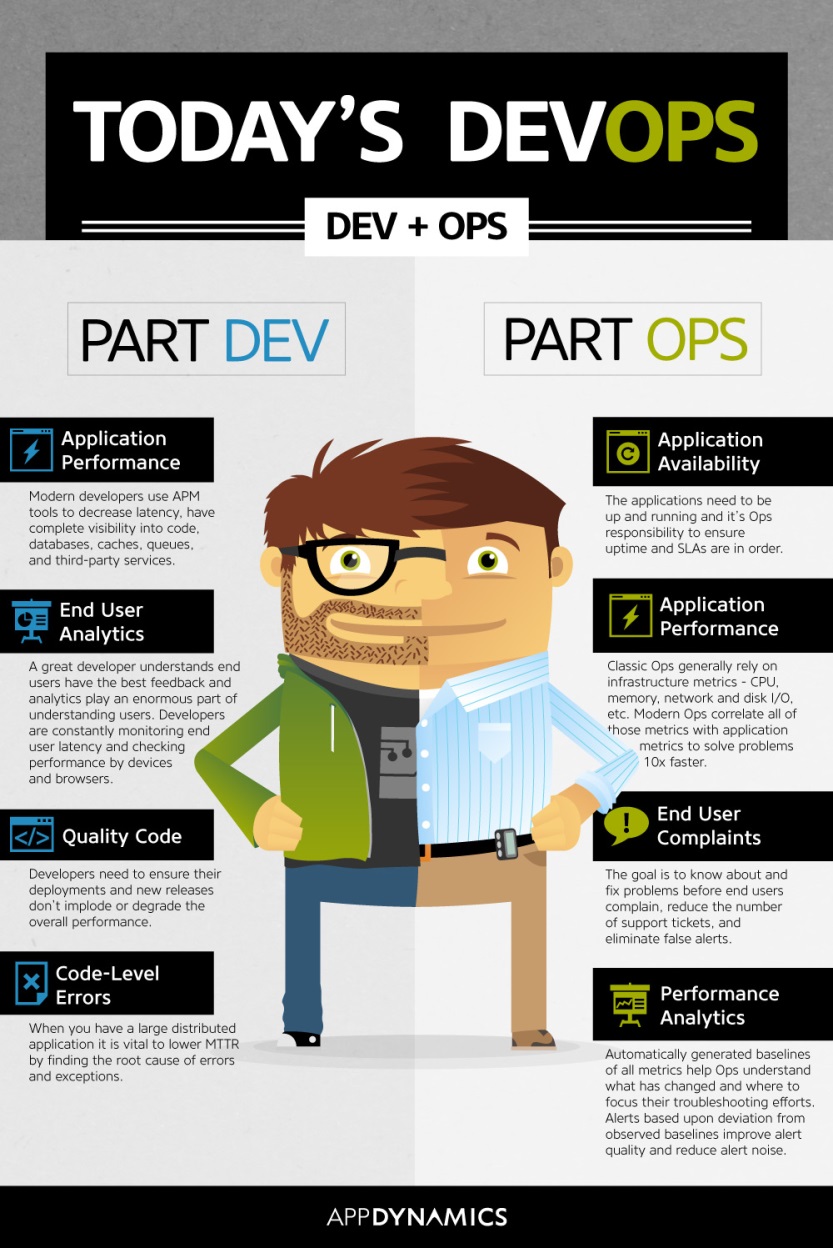
Le mouvement DevOps, contraction de « Development » et « Operational » tente de répondre à la problématique, certainement aussi ancienne que les DSI, qu’est la frontière qui sépare les développeurs (ceux qui écrivent le code source) et l’exploitation (ceux qui déploient et exploitent).

Cette problématique est remise au goût du jour notamment grâce à deux groupes d’acteurs qui sont les agilistes, qui ont amélioré

# DevOps mais encore?

Au vue de cette courte introduction, la première question que l’on ait amené à se poser est ; Pourquoi séparer Dev et Ops ?

## “The wall of confusion”



# Culture

## Features Team

“You build it, you run it”

## “Ops-ready”

## Anticiper l’activité Ops

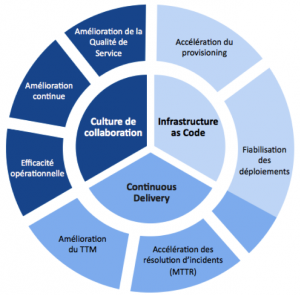
# Infrastructure as Code (Ops)

## Automatisation des environnements

(Puppet, Chef, …)

## Cloud computing

# Continuous delivery (livraison continue) (Dev)



## Continuous Integration (Intégration continue)

## Continuous Delivery (Livraison continue)

## Continuous Deployment (Déploiement continue)

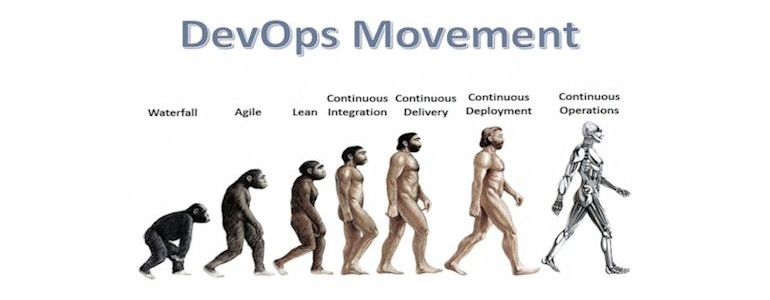
## Continuous Improvement

Etude de l’art

# Bluemix

# Axa

Plateforme d’Intégration Continue (PIC)



1. Introduction

# Définition du problème

De nos jours, de nombreuses méthodes de développement logiciel sont mises à la disposition des organisations informatique. Certaines d’entre elles ont émergé en raison des lacunes de leurs prédécesseurs et d’autres sont créées à partir de nouveaux concepts afin de répondre à une nouvelle philosophie de développement. Les organisations informatiques peuvent choisir librement l’une ou l’autre de ces méthodes en fonction des besoins liées à leurs projets.

Depuis maintenant quelques années (il est difficile de donner une date précise), les DSI s’appuient fortement sur la mouvance agile afin de mener à bien leurs projets. Aujourd’hui, les patterns agiles arrivent à maturité et offrent un éventail de méthodologies adaptables à tous les contextes. Les méthodes agiles garantissent la satisfaction du client et non la conformité aux termes d’un contrat de développement. Elles sont centrées sur la satisfaction de besoin du client et non sur les termes contractuels du projet. Nous n’allons pas aborder en profondeur le concept de l’agilité, ceci n’est pas le propos de ce mémoire, mais nous allons tout de même faire un petit rappel des idées fortes de cette méthodologie. Il faut des **cycles courts**, quelques semaines tout au plus, et découper le projet en **petites tâches** puis les **hiérarchiser** **en fonction du besoin**. Cela permet d’éviter le superflu et de se concentrer au début de chaque cycle sur ce qui a de la valeur pour l’utilisateur final. Le **feedback** permanent devient la règle d’or, avec des validations à chaque étape et des techniques ludiques d’évaluation de l’utilité des fonctions. L’agilité offre une meilleure visibilité et permet d’éviter les dérives observées lorsque les développeurs sont isolés. Le **changement** est autorisé voir encouragé, même tardivement,  car c’est un avantage décisif pour le client. Cela permet de ne pas se priver des bonnes idées en cours de route et surtout d’éliminer les mauvaises idées lancées au début du projet. Les méthodes agiles favorisent la **co-construction**, en intégrant l’annonceur lui-même dans le travail quotidien et en responsabilisant la totalité de l’équipe de développement, créant ainsi un véritable **esprit collaboratif** et l’ensemble du projet en gagne en qualité.

Cependant l’agilité, lorsqu’elle est exclusivement cantonnée au développement, se trouve néanmoins freinée par les tâches d’exploitation. Le mouvement DevOps a pour objectif d’étendre les pratiques agilistes à la livraison et au déploiement du projet.

Le mouvement DevOps, contraction de « Development » et « Operational » tente de répondre à la problématique, certainement aussi ancienne que les DSI, qu’est la frontière qui sépare les développeurs (ceux qui écrivent le code source) et l’exploitation (ceux qui déploient et exploitent).

« DevOps breaks the functional silos in the team to foster collaboration and focuses on business value by filling the gap between the development and operations departments. » (Michael Hüttermann, 2012)

« DevOps brise les silos fonctionnels dans l'équipe pour favoriser la collaboration et met l'accent sur la valeur de l'entreprise en comblant l'écart entre le développement et les opérations départements »

De nombreuses méthodes vont bien souvent être régies par des définitions et règles strictes entrainement de nombreux processus fondamentaliste. DevOps, maintenu par la communauté, laisse la place à votre propre interprétation. Par conséquent aucunes définition officielle de processus est disponible, bien que la théorie sous-jacente de DevOps d’appuie fortement sur la méthodologie agile. Le travail de recherche et de mise en place d’une culture DevOps doit être accès sur la façon dont l’entreprise peut l’intégrer dans leurs projets de développement Scrum.

# L’étude de cas

AXA

# Objectif et énoncé du problème

Ce mémoire de recherche a pour but d’identifier les différentes voies qui peuvent être choisies pour mettre en œuvre DevOps dans les entreprises s’appuyant sur les méthodes Scrum comme mode de développement et connaissant les problèmes cités plus haut. Nous allons tenter de proposer des chemins d’implémentations généraux afin d’aider les entreprises faisant face aux mêmes problèmes dans leurs processus de développement.

L’énoncé formel du problème de ce mémoire de recherche est formulé comme suit : Comment l’organisation informatique peut-elle prise en charge dans la mise en œuvre de DevOps ?

# Pertinence scientifique

Au cours de la dernière décennie, le terme de méthode d’ingénierie (ME) est apparu dans de nombreuses études scientifiques. Sjaak Brinkkemper l’a défini en 1996 comme suit:

« Method engineering is the scientific discipline that focuses on the design, construction and improvement of software development methods, techniques and tools. » (Brinkkemper, 1996)

« La méthode d’ingénierie est la discipline scientifique qui se concentre sur la conception, la construction et l’amélioration des méthodes de développement logiciels, des techniques et des outils. »

Depuis, la ME a beaucoup évolué et est devenue une discipline mature. La théorie et les techniques sous-jacentes permettent aux chercheurs et aux praticiens d’adapter les méthodes de développement des systèmes d’information (ISDMs) aux problèmes des entreprises.

# Pertinence business