MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE SUPÉRIEURE DE MANAGEMENT

POLE UNIVERSITAIRE KOLEA

MÉMOIRE DE FIN D'ÉTUDE

EN VUE DE L'OBTENTION D'UN MASTER ACADÉMIQUE EN

« MANAGEMENT STRATÉGIQUE ET SYSTÈME D'INFORMATION »

Vers un processus de développement logiciel optimisé : Analyse des pratiques et leviers d'amélioration

Cas : Naftal

Élaboré par :

- SAADADOU Zine eddinne Mohamed

Encadré par :

- Dr. DERRAR Hacene

Co-encadré par :

- Dr. LADJOUZI Soumia

Année 2024/2025

Résumé

Ce mémoire analyse le processus de développement logiciel chez Naftal dans une perspective d'optimisation. À travers une approche qualitative basée sur des entretiens, il met en lumière les limites des pratiques actuelles, marquées par l'absence de planification formelle, de documentation et d'outils de suivi. L'étude propose une hybridation entre les approches classiques et les méthodes agiles ; cette combinaison vise à structurer les projets tout en respectant les contraintes organisationnelles existantes. L'intégration d'une culture DevOps, de KPI, d'un pilotage par OKR et d'un accompagnement au changement permettrait de renforcer la performance, la coordination des équipes et l'alignement stratégique.

Mots clés : développement logiciel, optimisation des processus, méthodes agiles, documentation, formalisation, OKR, entreprise algérienne, gouvernance IT.

Abstract

This thesis analyzes the software development process at Naftal with a focus on optimization. Using a qualitative approach based on interviews, it highlights the limits of current practices, characterized by a lack of formal planning, documentation, and project tracking tools. The study proposes a hybridization between traditional approaches and agile methods to structure projects while considering existing organizational constraints. The integration of a DevOps culture, KPIs, OKR based management, and change support mechanisms is recommended to enhance performance, team coordination, and strategic alignment.

Keywords : software development, process optimization, agile methods, documentation, formalization, OKR, Algerian company, IT governance.

الملخص

حلّلت هذه المذكرة عملية تطوير البرمجيات في شركة نفطال بهدف تحسينها. ومن خلال مقاربة نوعية تعتمد على المقابلات، تبينت محدودية الممارسات الحالية التي تتسم بغياب التخطيط المنهجي، وتوثيق العمليات، وأدوات المتابعة. تقترح الدراسة اعتماد مزيج بين المناهج التقليدية والأساليب الرشيقة (Agile)، بهدف تنظيم المشاريع مع مراعاة القيود التنظيمية القائمة. كما توصي بإدماج ثقافة DevOps، ومؤشرات الأداء (KPI) ، ومنهجية OKR، إلى جانب مرافقة فعالة للتغيير، من أجل تعزيز الأداء، تنسيق الفرق وتحقيق التوافق الاستراتيجي.

الكلمات المفتاحية: تطوير البرمجيات، تحسين العمليات، الأساليب الرشيقة، التوثيق، التنظيم، OKR، المؤسسة الجزائرية، حوكمة نظم المعلومات.

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers Dieu, le Tout-Puissant, qui m’a accordé le courage et la patience nécessaires pour mener à bien ce travail.

Je dédie ce mémoire à mes chers parents et ma chère sœur, qui ont toujours cru en moi et m'ont soutenu à chaque étape difficile de ma vie. Leur amour et leurs encouragements ont été ma plus grande force.

À la mémoire de mon grand-père, dont les valeurs continuent de m’inspirer chaque jour.

À mes amis, pour leur présence, leur soutien et les moments de joie partagés.

J’adresse également mes sincères remerciements à mon tuteur de stage, Monsieur **ERRAHMANI**, pour son accompagnement et le temps précieux qu’il m’a consacré, à Madame **BELDIA** pour son accueil chaleureux et ses conseils avisés, ainsi qu'à Monsieur **MEKKAOUI** pour la confiance qu’il m’a accordée en me permettant d’effectuer mon stage au sein de la Direction centrale des systèmes d’information.

Je remercie aussi mes encadrants, Monsieur **DERRAR** et Madame **LADJOUZI**, pour leur guidance précieuse tout au long de ce travail.

Enfin, j’exprime toute ma reconnaissance envers mes professeurs, et toutes les personnes qui, de près ou de loin, m’a soutenu et contribué à l’élaboration de ce projet.

Table des matières

Liste des tableaux

Liste des figures

Liste des abréviations

Abréviation	Déscription
BPM	Business Process Management
BPMN	Business Process Model and Notation
CMMI	Capability Maturity Model Integration
DCSI	Direction Centrale des Systèmes d'Information
IA	Intelligence Artificielle
ITIL	Information Technology Infrastructure Library
OKR	Objectives and Key Results
SDLC	Software Development Life Cycle
SPA	Société Par Actions

INTRODUCTION GENERALE

Dans un contexte marqué par une transformation numérique rapide et une pression croissante sur les délais, les coûts et la qualité, les organisations sont confrontées à un défi de taille : concevoir et livrer des solutions logicielles performantes, fiables et évolutives tout en s'adaptant en permanence aux exigences du marché et aux besoins des utilisateurs. Le développement logiciel ne peut plus être envisagé comme une suite linéaire d'activités cloisonnées, mais comme un processus complexe, interdisciplinaire et évolutif, appelant des approches plus flexibles et optimisées. L'optimisation du processus de développement logiciel apparaît ainsi comme un levier stratégique pour améliorer la performance organisationnelle et garantir la satisfaction des parties prenantes.

Historiquement, les modèles traditionnels tels que le cycle en V ou le modèle en cascade ont constitué le socle méthodologique des projets informatiques. Ces approches basées sur une planification rigide et une exécution séquentielle ont permis de structurer les activités de développement et de renforcer la traçabilité des livrables. Toutefois, elles se sont progressivement révélées inadaptées aux environnements instables en raison de leur manque de réactivité et de leur faible capacité d'adaptation aux changements.

Face à ces défis, le Manifeste Agile, publié en 2001, a proposé un cadre méthodologique innovant centré sur la flexibilité et la collaboration. En privilégiant la valeur apportée au client, l'adaptation aux changements et l'autonomie des équipes, les méthodes agiles ont transformé les pratiques du développement logiciel. Des cadres comme Scrum, Extreme Programming (XP) ou DevOps ont été largement intégrés par les entreprises à la recherche d'innovation, valorisant le travail en itérations courtes, la communication transparente et l'accélération des livraisons. Parallèlement, des modèles de maturité comme le CMMI ont offert des référentiels normés pour organiser, évaluer et améliorer les processus à l'échelle globale de l'organisation. Plutôt que de s'opposer, ces approches se complètent souvent, comme le montre l'exemple du Systematic Software Engineering où l'association de Scrum et du CMMI a permis d'allier exigence méthodologique et agilité opérationnelle.

L'optimisation du processus de développement logiciel repose ainsi sur une synergie entre structuration et adaptabilité. Elle nécessite une compréhension fine des enjeux organisationnels, des contraintes techniques, des dynamiques humaines, mais aussi des leviers d'amélioration disponibles. La gestion par processus, notamment à travers la cartographie, l'identification des points faibles et la mise en place d'indicateurs de performance fournit un cadre structurant indispensable. Parallèlement, l'agilité introduit une

capacité d'expérimentation et de révision continue qui permet de rendre les processus plus réactifs, plus collaboratifs et plus centrés sur la valeur. Cette hybridation méthodologique est aujourd'hui au cœur des pratiques les plus performantes dans le domaine du génie logiciel.

Ce mémoire s'inscrit dans cette dynamique en proposant une analyse approfondie des pratiques actuelles de développement logiciel chez Naflal. L'objectif est d'identifier les conditions organisationnelles, méthodologiques et culturelles qui favorisent un développement logiciel efficient, mais aussi de comprendre les freins et résistances qui peuvent entraver l'optimisation du processus de développement logiciel. La réflexion s'appuie sur une revue critique des modèles classiques et agiles, des outils de gouvernance comme les OKR, des cadres de maturité (CMMI) ainsi que des expérimentations menées dans des contextes variés, y compris dans des organisations publiques algériennes. À travers cette démarche, il s'agira d'apporter des recommandations pratiques et un cadre méthodologique adapté au contexte de Naftal pour penser et mettre en œuvre un processus de développement logiciel véritablement optimisé, au service de la performance et de la satisfaction client.

Problématique

Pour délimiter le sujet, l'interrogation principale à laquelle ce mémoire tente de répondre est formulée de la manière suivante :

"Comment formaliser de manière progressive le processus de développement logiciel au sein de la DCSI de Naftal, afin d'en améliorer la performance et la coordination ?"
De cette problématique principale découlent les questions secondaires suivantes :

- Quels leviers organisationnels, humains et technologiques mobiliser pour initier une transformation agile dans un contexte peu structuré ?
- Dans quelle mesure la gouvernance par les Objectives and Key Results (OKR) peut-elle renforcer l'alignement entre les objectifs stratégiques et les actions opérationnelles ?
- Comment la mise en place d'indicateurs de performance peut-elle contribuer à professionnaliser la gestion des projets informatiques dans un contexte organisationnel peu structuré ?

- Comment favoriser l’adhésion des équipes dans un projet de transformation numérique impliquant un changement des pratiques établies ?

Organisation de mémoire :

Afin de fournir des éléments de réponse la problématique exposée, ce mémoire a été structuré de la manière suivante :

Le premier chapitre, intitulé " Revue de littérature et Cadre conceptuel ", aborde les travaux principaux sur lesquels repose cette recherche, ainsi que le cadre conceptuel qui présente les définitions des différents concepts en relation avec le sujet.

Dans le deuxième chapitre, "Cadre méthodologique et Organisationnel" présente l’organisme d’accueil ainsi que le cadre méthodologique suivi pour répondre à la question de recherche.

Enfin, le troisième chapitre est consacré aux "Résultats et Discussion" il va présenter les résultats de recherche et les propositions tirées de ces résultats.

CHAPITRE I : REVUE DE LITTERATURE ET CADRE CONCEPTUEL

Section 1 : Revue de littérature

Dans cette section dédiée à la revue de la littérature, plusieurs réflexions vont être présentées que ça soit en relation avec le développement logiciel, la gestion des processus, ou tout ce qui est bonnes pratiques et référentiels.

1.1. L'approche processus

L'approche processus est unanimement reconnue pour son rôle dans l'amélioration de l'efficacité organisationnelle en modélisant les activités internes. Selon plusieurs auteurs, cette approche permet de structurer les activités d'une organisation de manière cohérente et de faciliter leur gestion. (Aissaoui, et al., 2022) Ainsi que (Serehane & Talbi, 2015) soulignent l'importance de cette méthode dans le cadre de la gestion des processus opérationnels, sans oublier la norme ISO 9001 qui est fréquemment citée comme un cadre de référence permettant de garantir la qualité des processus et d'assurer leur conformité.

Cependant, les avis divergent fortement quant à la flexibilité des méthodes classiques, certaines analyses les jugeant trop rigides tandis que d'autres soulignent leur efficacité dans des contextes bien structurés.

D'une part, l'article de (Charbi & Guesmi, 2021) met en évidence les limites du Business Process Management (BPM) et des méthodes structurées traditionnelles en termes de rigidité tout en soulignant les apports des approches agiles, jugées plus souples et mieux adaptées aux environnements jugés instable.

D'autre part, les recherches de (Aissaoui, et al., 2022) et de (Serehane & Talbi, 2015) valorisent la structuration apportée par les approches classiques en mettant l'accent sur leur efficacité dans des contextes bien définis.

Par ailleurs, il est important de souligner que la méthodologie adoptée par (Serehane & Talbi, 2015) reste théorique et manque de validation empirique, contrairement à l'approche plus pratique de la recherche-action utilisée par (Aissaoui, et al., 2022).

1.2. L'importance de la gestion de processus

La gestion des processus est perçue comme un levier essentiel pour l'optimisation organisationnelle, notamment en ce qui concerne la standardisation des pratiques et la

réduction des coûts. Selon (Aissaoui, et al., 2022) et (Toumi Amara & Kouloughli, 2021), une gestion efficace des processus permet de garantir une meilleure uniformité des pratiques au sein d'une organisation et contribue directement à l'amélioration de la performance. De plus, l'utilisation d'indicateurs de performance s'avère cruciale afin d'évaluer la performance des processus et d'adapter les orientations stratégiques en fonction des résultats observés, comme le mentionnent (Serehane & Talbi, 2015).

Toutefois, plusieurs contradictions apparaissent dans cette littérature. Premièrement, le focus sectoriel de certaines études limite leur portée pendant que (Aissaoui, et al., 2022) se concentrent sur le secteur de la santé, (Toumi Amara & Kouloughli, 2021) abordent principalement la gestion des ressources humaines, ce qui rend difficile la généralisation des résultats à d'autres secteurs. De plus, l'étude de (Serehane & Talbi, 2015) ne quantifie pas l'impact des indicateurs de performance qu'ils proposent, ce qui laisse une zone d'incertitude quant à leur efficacité dans la pratique.

1.3. La gestion des processus en informatique

L'utilisation des outils de modélisation de processus comme le Business Process Model and Notation (BPMN) et des logiciels tels que Bonita est un sujet récurrent dans la gestion des processus en informatique. (Toumi Amara & Kouloughli, 2021) Ainsi que (Aissaoui, et al., 2022) mettent en évidence que l'automatisation des processus constitue un atout majeur de ces outils dans la mesure où l'efficacité opérationnelle est améliorée et les erreurs humaines sont réduites. Cependant, des contradictions apparaissent lorsque l'on aborde l'intégration des méthodes agiles dans ce domaine. (Charbi & Guesmi, 2021) Privilégient une approche agile, favorisant l'adaptabilité et l'itération rapide tandis que (Toumi Amara & Kouloughli, 2021) se basent sur des méthodes plus classiques de modélisation comme le BPMN sans intégrer la flexibilité des approches agiles. En outre, aucun des articles étudiés n'aborde de manière approfondie les défis posés par l'intégration des technologies émergentes telles que l'intelligence artificielle (IA) ou le Big Data dans la gestion des processus, ce qui représente une limite importante dans la compréhension des enjeux technologiques actuels.

1.4. La contribution des méthodes agiles

Les méthodes agiles sont largement reconnues pour leur capacité à améliorer l'adaptabilité des organisations particulièrement dans des environnements dynamiques et complexes. (Charbi & Guesmi, 2021) Mettent en évidence que ces méthodes permettent une gestion plus souple des projets, réduisant les risques par des itérations courtes et une réévaluation continue des priorités. Cependant, la pertinence des méthodes agiles reste limitée à certains types de projets. L'étude de (Charbi & Guesmi, 2021) suggère que l'agilité est plus appropriée aux projets caractérisés par une forte variabilité, tandis que (Aissaoui, et al., 2022) défendent l'idée que le BPM demeure pertinent pour les processus plus stables comme illustré dans leur étude sur le secteur de la santé. De plus, une limite importante des méthodes agiles réside dans leur application à des organisations ayant une forte culture de résistance au changement. Là où (Serehane & Talbi, 2015) insistent sur la nécessité de gérer les compétences et les résistances au sein des équipes, l'article de (Charbi & Guesmi, 2021) ne prend pas en compte ces aspects humains, ce qui peut nuire à l'adoption des méthodes agiles dans certains contextes organisationnels.

1.5. Le management agile

Le management agile s'est imposé au début des années 2000 comme une réponse aux limites des approches traditionnelles de gestion, jugées souvent rigides et peu adaptées aux environnements instables. Construit autour de principes de flexibilité, de réactivité, de collaboration et d'amélioration continue, le management agile est aujourd'hui considéré comme un levier stratégique pour optimiser les processus en générale et dans le développement logiciel en particulier (Horney, 2013) cité dans (Boukhedimi, Zerrouki, & Merad, 2023).

D'un point de vue organisationnel, l'agilité repose sur des équipes auto-organisées, multidisciplinaires et autonomes qui sont capables d'adapter leur mode de fonctionnement en fonction du contexte (Barrand, 2009) Cité dans (Boukhedimi, Zerrouki, & Merad, 2023). Contrairement aux modèles séquentiels, le management agile valorise une approche itérative et incrémentale, où chaque cycle de développement court permet un retour rapide du client et ainsi une amélioration continue.

Sur le plan stratégique, l'agilité est souvent perçue comme une compétence organisationnelle indispensable dans un environnement décrit comme VUCA : Volatile, Incertain, Complexe et Ambigu (Bennett & Lemoine, 2014) cité dans (Boukhedimi, Zerrouki, & Merad, 2023). Elle permet aux entreprises d'anticiper les évolutions du marché, de reconfigurer leurs processus de manière proactive et de mieux aligner leurs décisions opérationnelles sur les attentes clients (Sanchez, 2004) cité dans (Boukhedimi, Zerrouki, & Merad, 2023).

Les avantages du management agile sont nombreux, il contribue à réduire les délais de livraison, améliore la qualité des produits en intégrant des retours fréquents, stimule l'innovation par l'expérimentation et favorise l'engagement des équipes grâce à une culture fondée sur la confiance et l'autonomie (Boukhedimi, Zerrouki, & Merad, 2023). Ces avantages participent directement à l'optimisation du processus de développement logiciel, en assurant une meilleure gestion du temps, une réduction des pertes financières et une orientation plus forte vers la valeur livrée au client.

Toutefois, le passage à l'agilité nécessite des transformations culturelles importantes. Des freins peuvent émerger, notamment dans les organisations à forte hiérarchie. L'instauration de pratiques agiles suppose une redéfinition des responsabilités, et une gestion du changement adaptée (Grosjean, 2003) cité dans (Boukhedimi, Zerrouki, & Merad, 2023).

Ainsi, le management agile, bien qu'issu du domaine informatique, tend à devenir un modèle de référence en matière de gestion de projets dans des environnements dynamiques. Son application dans le développement logiciel en fait un outil central pour l'optimisation des processus grâce à sa capacité à conjuguer performance, flexibilité et orientation client.

1.6. Lien entre la gestion des processus et l'agilité

Dans les environnements numériques et technologiques, l'agilité devient un facteur d'adaptation des processus eux-mêmes. Comme le souligne (Boukhedimi, Zerrouki, & Merad, 2023), les méthodes agiles n'abolissent pas les processus, mais les rendent plus légers, collaboratifs, adaptables, et centrés sur la valeur ajoutée pour le client. La gestion par processus fournit ainsi un cadre structurant nécessaire à la coordination des activités, tandis que l'agilité instaure une dynamique de flexibilité permettant de réajuster le fonctionnement des processus en fonction des retours d'expérience, des contraintes et de l'évolution des priorités du projet.

La synergie entre processus et agilité s'opère notamment à travers une priorisation des tâches par valeur, une organisation du travail en itérations courtes nommé « sprints », et des rétrospectives régulières permettant l'ajustement du processus en continu. Cela implique une redéfinition de la démarche processus : il ne s'agit plus uniquement d'optimiser des chaînes d'activités fixes, mais de rendre les processus eux-mêmes évolutifs, ce que certains auteurs nomment « processus agiles » (Etasse & Rousseau, 2023) cité par (Boukhedimi, Zerrouki, & Merad, 2023).

Par ailleurs, dans les organisations modernes, le management agile valorise l'autonomie des équipes dans la gestion de leurs micro-processus, tout en s'inscrivant dans une logique d'alignement avec les objectifs stratégiques globaux. La combinaison entre la vision macroscopique apportée par la gestion des processus, et la souplesse tactique des méthodes agiles, permet ainsi une optimisation simultanée des dimensions structurelles et fonctionnelles.

Cependant, cette hybridation suppose des conditions préalables : une culture du changement, une gouvernance participative, et des outils de suivi adaptés (comme le BPMN, les outils de collaboration ou de visualisation agile). Sans cela, le risque est de retomber dans une impasse organisationnelle, tiraillée entre une bureaucratie paralysante et un chaos opérationnel dû à l'absence de cadre structurant.

En définitive, l'agilité ne remplace pas la gestion des processus mais elle la transforme. Elle en renouvelle les finalités en mettant l'accent sur l'expérimentation, la co-construction et l'adaptation continue, contribuant ainsi à une approche plus organique de l'optimisation des processus dans les environnements complexes.

1.7. La gestion agile dans les entreprises publiques Algériennes : le cas de Algérie télécom

L'adoption de pratiques agiles dans les organisations publiques algériennes reste encore marginale, bien que certaines initiatives témoignent d'une volanté de transformation. L'étude menée par (Boukhedimi, Zerrouki, & Merad, 2023) met en lumière les opportunités, les obstacles et les premiers résultats liés à la mise en œuvre du management agile dans ce contexte spécifique.

L'exemple de Sonelgaz, entreprise publique du secteur de l'énergie, illustre une volonté d'expérimentation des méthodes agiles dans la gestion de projets liés aux énergies renouvelables. La constitution d'une équipe dédiée, travaillant selon les principes de Scrum (collaboration, itérations courtes, livrables réguliers), représente une tentative de réorganiser les processus de pilotage et d'accroître la réactivité face à des enjeux technologiques et environnementaux évolutifs. Cette initiative a notamment favorisé une meilleure transparence des activités, une implication plus importante des membres de l'équipe, et une coordination facilitée des tâches (Boukhedimi, Zerrouki, & Merad, 2023).

L'étude cite également Algérie Télécom, acteur majeur du secteur des télécommunications, qui s'est engagé dans une série d'initiatives traduisant une orientation vers l'agilité. Parmi ces initiatives :

- Ouverture d'un centre d'appel centré sur le client,
- Introduction de paiements électroniques,
- Création d'une boutique virtuelle,
- Lancement d'un concours d'innovation destiné à stimuler la créativité interne et externe.

Ces actions s'inscrivent dans une stratégie d'innovation continue, en cohérence avec les principes agiles de proximité client, expérimentation rapide et amélioration incrémentale. Elles montrent également une capacité à réagir aux changements du marché et à digitaliser les services, dans un secteur où la technologie progresse sans cesse.

Cependant, les résultats de l'étude soulignent aussi plusieurs freins structurels à l'agilité dans les entreprises publiques algériennes :

- Une culture fortement hiérarchisée, s'oppose aux principes de l'auto-organisation.
- Une faible autonomie des équipes, souvent soumises à des directives descendantes,
- Un manque de formation aux pratiques agiles,
- Une résistance au changement, en particulier dans les services fonctionnels.

En outre, les processus décisionnels restent encore lents et bureaucratiques, ce qui limite la réactivité et empêche souvent une adaptation rapide aux besoins des clients. La logique agile,

qui repose sur la responsabilisation, l'adaptabilité et la collaboration horizontale, entre donc en tension avec les logiques de contrôle héritées des modèles organisationnels traditionnels.

Malgré ces obstacles, l'étude conclut que l'agilité représente un levier stratégique de transformation, à condition de repenser les pratiques managériales, d'investir dans la formation, et de développer une culture d'amélioration continue. Des initiatives comme celles d'Algérie Télécom peuvent servir de références locales inspirantes, à condition d'être consolidées et diffusées à plus large échelle.

1.8. L'approche Scrum et DevOps : deux piliers complémentaires pour l'agilité

Scrum s'impose comme l'une des méthodes agiles les plus répandues dans les organisations de développement logiciel. Basée sur un processus itératif et incrémental la méthode Scrum vise à maximiser la valeur délivrée au client tout en permettant une adaptation continue aux exigences changeantes (Schwaber & Sutherland, 2020). La structure de Scrum repose sur trois rôles clés : Product Owner, Scrum Master et l'équipe de développement ainsi que sur des artefacts et cérémonies : sprint, backlog, revue de sprint, rétrospective, qui permettent un pilotage efficace et transparent du projet. Le principe de responsabilisation collective et de collaboration étroite avec les parties prenantes incarne directement les valeurs du Manifeste Agile (Beck, et al., 2001), telles que la collaboration avec le client plus que la négociation contractuelle ou l'adaptation au changement plus que le suivi d'un plan.

En complément, DevOps propose une approche intégrée du développement et des opérations, favorisant l'automatisation, l'intégration continue (CI), et le déploiement continu (CD). Cette synergie entre développeurs et opérationnels, souvent négligée dans les organisations traditionnelles, contribue à réduire les cycles de mise en production tout en renforçant la fiabilité des livrables (Amershi, et al., 2019). Dans un contexte où les projets sont de plus en plus complexes, l'association Scrum et DevOps se révèle particulièrement efficace pour concilier innovation rapide, stabilité des systèmes et satisfaction client.

1.9. Repenser la structure organisationnelle pour soutenir l'agilité

L'un des obstacles majeurs à l'efficacité des méthodes agiles réside dans la structuration traditionnelle des équipes, souvent cloisonnées en silos fonctionnels. Cette organisation limite la transversalité et entrave la communication entre les acteurs du projet. Pour y

remédier, la littérature recommande une réorganisation autour d'équipes multidisciplinaires et autonomes, capables de gérer l'ensemble du cycle de développement d'un produit.

(Amershi, et al., 2019) Insistent sur la nécessité de structurer les équipes autour des produits, plutôt que des fonctions, afin d'encourager une responsabilisation accrue, une meilleure appropriation des objectifs et une réduction des dépendances inter-équipes. Dans cette logique, chaque équipe devient responsable non seulement du développement, mais aussi de la qualité, du déploiement et de la maintenance des solutions. Cette approche permet d'instaurer un véritable esprit de collaboration, en cohérence avec les valeurs du Manifeste Agile, notamment l'interaction entre individus et la réactivité face au changement.

Néanmoins, cette transformation structurelle ne peut s'opérer sans une conduite du changement appropriée. La résistance culturelle, la difficulté à redéfinir les rôles et l'absence de vision commune sont autant de freins potentiels à la mise en œuvre d'une organisation agile. Une approche progressive et itérative, soutenue par une communication transparente, s'avère donc indispensable pour réussir cette transition.

1.10. Mettre en place un cadre de gouvernance agile : l'exemple des OKR

L'un des défis majeurs de l'agilité à l'échelle organisationnelle est la cohérence stratégique entre les équipes autonomes. Pour répondre à ce besoin, la mise en place d'un cadre de gouvernance agile basé sur les Objectives and Key Results (OKR) constitue une piste prometteuse. Cet outil, initialement popularisé par Google, permet d'aligner les objectifs des équipes avec la vision globale de l'entreprise tout en laissant une marge d'autonomie sur les moyens à mettre en œuvre.

Les OKR s'inscrivent pleinement dans une logique agile car ils sont définis sur des cycles courts qui sont généralement trimestriels, sont mesurables, transparents et réévalués régulièrement. Contrairement aux indicateurs de performance traditionnels, ils ne se contentent pas de mesurer l'efficacité mais visent également à stimuler l'innovation, en incitant les équipes à se fixer des objectifs ambitieux et à expérimenter de nouvelles approches. (Amershi, et al., 2019) Évoquent d'ailleurs la nécessité d'introduire des métriques orientées vers l'impact utilisateur, la collaboration interdisciplinaire et la capacité d'itération, ce qui rejoint l'esprit des OKR.

Malgré leur potentiel les OKR posent également certaines limites. Une mauvaise définition des objectifs ou un manque d'appropriation par les équipes peut conduire à une perte de sens ou à une focalisation excessive sur la performance quantitative. Il est donc essentiel d'accompagner la mise en place des OKR d'une réflexion sur la culture organisationnelle, la formation des managers et l'implication des collaborateurs dans la définition des résultats clés.

1.11. L'importance du CMMI et sa contribution dans l'optimisation des processus logiciels : le cas de Systematic

Le modèle Capability Maturity Model Integration (CMMI) occupe une place centrale dans le champ de l'ingénierie logicielle, il représente un outil stratégique d'amélioration continue des processus. Développé par le Software Engineering Institute, ce modèle propose une approche structurée permettant aux organisations d'évaluer leur niveau de maturité organisationnelle et de piloter de manière proactive la performance de leurs projets. Le modèle CMMI est important car il permet d'introduire une méthodologie rigoureuse dans l'organisation, tout en offrant une vision globale de la qualité et de la gestion des risques.

Dans le cadre de l'étude menée par (Sutherland, Jakobsen, & Johnson, 2009), l'exemple de l'entreprise « Systematic Software Engineering » illustre de façon claire la portée concrète du CMMI dans l'optimisation du développement logiciel. Systematic Software Engineering, évaluée au niveau 5 du CMMI, a su capitaliser sur ce modèle pour atteindre un niveau de maîtrise avancé de ses processus internes. Ce niveau 5 représente le stade le plus abouti de maturité, où l'organisation adopte une démarche d'amélioration continue, fondée sur la mesure, l'analyse et l'optimisation systématique des résultats issus des projets.

La contribution du CMMI dans ce contexte se manifeste d'abord par la capacité à réduire significativement les défauts et les erreurs de code. Chez Systematic Software Engineering, l'application rigoureuse des processus CMMI a permis de diminuer de 42 % la quantité de code à retravailler. D'un autre point de vue, 92 % des jalons de projets étaient livrés en temps ou en avance, preuve d'une gestion maîtrisée des délais. Ces résultats ne relèvent pas de mesures isolées, ils sont issus d'une organisation où les données projet sont systématiquement collectées, analysées et comparées à des référentiels de performance. Ce pilotage par les données permet d'objectiver les écarts, de mieux anticiper les risques et de garantir une meilleure prévisibilité des résultats.

En instaurant un référentiel unifié de processus applicable à l'ensemble de ses projets, Systematic Software Engineering a également renforcé la transversalité et la mutualisation des expériences. Les équipes peuvent passer d'un projet à l'autre sans perdre en efficacité, car les standards sont partagés. Cette homogénéisation favorise l'apprentissage organisationnel, un élément fondamental dans une logique d'amélioration continue. En parallèle, la standardisation des processus, combinée à une discipline rigoureuse dans leur exécution, a permis de réduire la charge globale de travail à 69 % de celle observée dans des entreprises non certifiées CMMI. Cela signifie que la qualité a été améliorée tout en réduisant l'effort consenti, ce qui témoigne d'un gain d'efficience non négligeable.

Un autre aspect déterminant réside dans la reconnaissance externe qu'offre une certification CMMI de haut niveau. Pour Systematic Software Engineering cela a représenté un levier stratégique dans la conquête de nouveaux marchés, notamment dans les domaines de la défense et de la santé où la fiabilité des systèmes est cruciale. Les clients de ces secteurs valorisent particulièrement la capacité à livrer des solutions robustes, maintenables et adaptables, dans des délais maîtrisés. Ainsi, CMMI agit également comme un gage de crédibilité et de confiance, renforçant la position concurrentielle de l'entreprise sur des appels d'offres.

Enfin, au-delà de la conformité aux standards, la démarche CMMI a permis à Systematic Software Engineering de développer une culture de la qualité profondément ancrée dans ses pratiques quotidiennes. L'organisation ne se contente pas d'exécuter des processus ; elle les évalue, les remet en question et les adapte en fonction des retours du terrain. Cette posture constitue l'essence même du niveau 5 du CMMI, où chaque projet devient une opportunité d'apprentissage et d'amélioration.

En définitive, l'importance de CMMI dans le cas de Systematic Software Engineering ne se limite pas à l'atteinte d'un label ou à la mise en œuvre de bonnes pratiques. Elle amene l'organisation à devenir un système capable de s'auto-évaluer, d'apprendre de ses expériences et de s'adapter en permanence. L'expérience de Systematic Software Engineering démontre que loin d'être un cadre rigide, CMMI peut être un puissant moteur d'excellence, à condition d'être intégré dans une démarche de progrès authentique et portée par l'ensemble de l'organisation.

1.12. La complémentarité entre CMMI et Scrum : rigueur et agilité au service de la performance

Longtemps perçus comme appartenant à deux approches méthodologiques distinctes et parfois opposées, CMMI et Scrum sont pourtant étudiés ensemble dans les travaux de (Sutherland, Jakobsen, & Johnson, 2009). À travers l'analyse du cas de l'entreprise Systematic Software Engineering, les auteurs proposent une lecture novatrice qui dépasse l'opposition classique entre méthodes structurées et méthodes agiles. Ils démontrent qu'il est non seulement possible, mais surtout profitable de combiner CMMI et Scrum dans une logique d'optimisation des processus.

Le modèle CMMI, en tant que référentiel de maturité des processus repose sur la mise en place de pratiques rigoureuses, documentées et mesurables. Il vise à garantir la stabilité, la prévisibilité et la qualité des livrables, en s'appuyant sur une démarche d'amélioration continue à l'échelle de l'organisation. À l'inverse, Scrum s'inscrit dans une approche agile centrée sur l'adaptation rapide au changement, la collaboration avec le client et les livraisons fréquentes de versions opérationnelles du produit. Bien que la combinaison de ces deux approches puisse paraître contre-nature au départ, l'expérience de Systematic Software Engineering prouve qu'elles forment en réalité une complémentarité stratégique.

Dans les projets pilotes menés chez Systematic Software Engineering l'introduction progressive de Scrum dans un environnement déjà certifié CMMI niveau 5 a permis de concilier discipline des processus et agilité opérationnelle. Scrum n'a pas remplacé les pratiques existantes, mais les a enrichies et dynamisées. Il a permis d'accélérer les cycles de développement, de renforcer la transparence avec les clients et d'améliorer la communication au sein des équipes. En intégrant les principes de Scrum tels que les sprint et réunions quotidiennes dans une organisation déjà habituée à la mesure et à l'analyse des performances, Systematic Software Engineering a pu augmenter sa réactivité tout en maintenant un haut niveau de qualité.

Cette complémentarité se traduit par des résultats concrets : la productivité des projets a été multipliée par deux dans certains cas, tandis que les défauts détectés lors des phases de tests finaux ont été réduits de 38 à 42 %. De plus, la collaboration avec les clients est devenue encore plus efficacité. Grâce aux livraisons itératives et à la participation active des parties prenantes, il est devenu possible de réajuster les priorités en cours de projet, d'identifier

rapidement les problèmes techniques et de recentrer le développement sur les véritables besoins des utilisateurs.

Dans le cas de Systematic Software Engineering, CMMI a permis de sécuriser et de structurer l'introduction de Scrum. Inversement, Scrum a offert à l'organisation un levier pour accélérer l'innovation, renforcer la collaboration avec les clients et améliorer la satisfaction des utilisateurs. Les deux approches se renforcent mutuellement, CMMI garantit que l'ensemble des processus pertinents sont bien pris en compte, tandis que Scrum assure qu'ils sont mis en œuvre de manière efficace, rapide et adaptable.

Ce rapprochement méthodologique ouvre des perspectives intéressantes, notamment pour les organisations opérant dans des secteurs exigeants où les contraintes de qualité, de sécurité ou de conformité réglementaire sont fortes. Il montre qu'il est possible d'allier rigueur et souplesse, et que l'agilité ne doit pas être synonyme d'improvisation, tout comme la formalisation ne doit pas exclure la réactivité.

Pour éclaircir la différence entre un cadre référentiel centré sur les processus tel que CMMI, et un cadre méthodologique agile centré sur l'équipe et la flexibilité comme Scrum, le tableau suivant résume les points essentiels à retenir :

Tableau 1 : La comparaison entre CMMI et Scrum

Critère	CMMI	Scrum
Objectif	Améliorer la maturité des processus organisationnels	Améliorer la collaboration et l'adaptabilité des équipes
Approche	Planifiée, structurée, normative	Itérative, empirique, adaptative
Portée	Organisationnelle, concerne toute l'entreprise	Équipe de projet, focalisé sur le produit
Pratiques clés	Processus définis, mesures, audits, amélioration continue	Backlog, sprints, revues, rétrospectives
Rôles	Plutôt orienté management et process owner	Product Owner, Scrum Master, équipe Scrum

Adaptabilité	Moins flexible (suivi rigoureux des niveaux de maturité)	Très flexible (adaptation constante)
Orientation	Processus et qualité à l'échelle organisationnelle	Collaboration et livraison à l'échelle équipe
Niveau visé	5 niveaux (Initial à Optimisé)	Pas de niveaux, cycle d'amélioration continue

Source : Elaboration personnelle à partir de (Sutherland, Jakobsen, & Johnson, 2009)

En somme, l'expérience de Systematic Software Engineeri ng témoigne d'une intégration harmonieuse entre deux visions complémentaires du développement logiciel. Loin d'opposer CMMI et Scrum, les auteurs plaident pour une approche intégrée, où la rigueur méthodologique soutient l'agilité terrain, et où l'agilité vient nourrir l'amélioration continue des processus.

1.13. Conclusion de la revue de littérature

Les précédents titres ont permis de dresser un panorama des concepts fondamentaux liés à l'approche processus et aux méthodes agiles tout en explorant leur articulation dans les environnements organisationnels contemporains. Il apparaît que la gestion par processus constitue un cadre structurant indispensable à la coordination et à l'optimisation des activités (Aissaoui, et al., 2022) et (Serehane & Talbi, 2015), tandis que l'agilité introduit une dynamique de flexibilité et d'adaptation particulièrement précieuse dans les environnements incertains (Charbi & Guesmi, 2021) et (Boukhedimi, Zerrouki, & Merad, 2023).

Les études examinées révèlent néanmoins des limites méthodologiques et sectorielles, qui restreignent parfois la portée généralisable des résultats (Toumi Amara & Kouloughli, 2021). Par ailleurs, si des initiatives agiles émergent dans les organisations publiques algériennes, elles se heurtent encore à des freins culturels et structurels (Boukhedimi, Zerrouki, & Merad, 2023). Enfin, les approches hybrides telles que Scrum couplé à DevOps ouvrent des perspectives prometteuses pour une agilité à la fois opérationnelle et technologique (Schwaber & Sutherland, 2020) et (Amershi, et al., 2019).

En définitive, ce chapitre souligne que l'optimisation des processus de développement logiciel ne peut se concevoir sans une intégration fine entre rigueur structurelle et souplesse organisationnelle, un équilibre dont les organisations doivent tenir compte pour évoluer efficacement dans des environnements complexes.

Section 2 : Cadre conceptuel

Le cadre conceptuel constitue la base théorique de ce mémoire. Il va permettre de définir les notions clés, de structurer la réflexion et d'établir les liens entre les différentes variables évoqués dans la revue de littérature.

2.1. Définition générale du processus

Dans le cadre de la gestion des organisations et des systèmes d'information, un processus se définit comme un enchaînement structuré et corrélé d'activités visant à transformer des éléments d'entrée en un résultat attendu, qu'il s'agisse d'un produit ou d'un service. Selon la norme ISO 9000 : 2015 un processus est « un ensemble d'activités corrélées ou en interaction qui utilise des éléments d'entrée pour produire un résultat escompté », ces éléments peuvent être matériels ou informationnels (Toumi Amara & Kouloughli, 2021).

D'un point de vue opérationnel, le processus suppose des entrées mesurables, une valeur ajoutée produite au travers de l'enchaînement des tâches, des sorties mesurables, ainsi que la possibilité de réitération (Cattan, Idrissi, & Knockaert, 1998) cité par (Toumi Amara & Kouloughli, 2021). Il mobilise différents moyens (ressources humaines, équipements, procédures, etc.) pour atteindre un objectif déterminé.

Il est aussi important de souligner que le terme processus est souvent confondu avec les notions de procédure et de procédé, bien qu'ils renvoient à des réalités distinctes. Tous trois partagent une origine commune, le latin procedere, signifiant « aller de l'avant » et impliquent une certaine progression chronologique (Morley, Bia-Figueiredo, & Gillette, 2011). Toutefois, leurs usages ont évolué différemment selon les contextes.

La procédure désigne une manière spécifiée et formalisée d'exécuter un processus. Elle se distingue par son ancrage organisationnel : les rôles y sont définis, les outils identifiés (documents, logiciels, etc.), et les méthodes de travail fixées. Ainsi, un même processus peut être décliné en plusieurs procédures selon les contextes ou les organisations (Morley, Bia-Figueiredo, & Gillette, 2011). En ce sens, la procédure représente la traduction opérationnelle d'un processus au sein d'une structure formalisée.

Quant au procédé, il fait davantage référence à une méthode technique ou scientifique utilisée pour atteindre un résultat donné, notamment dans les domaines industriels. Il se

rapproche d'une manière de faire, souvent standardisée, utilisée dans la production ou la fabrication.

Pour résumer, si le processus désigne l'ensemble des activités créatrices de valeur, La procédure représente la traduction formalisée d'une action selon des normes définies dans un cadre organisationnel, tandis que le procédé relève d'une méthode technique appliquée à un champ précis.

2.2. Le cycle de vie du développement logiciel

Le cycle de vie du développement logiciel ou Software Development Life Cycle (SDLC) en anglais est un processus structuré que les équipes de développement utilisent pour concevoir et produire des logiciels de qualité. Il repose sur une série d'étapes qui divisent le développement en tâches précises afin de faciliter leur gestion, leur réalisation et surtout leur suivi (Amazon Web Services, s.d.).

Le cycle de vie du développement logiciel constitue un cadre méthodologique qui permet de gérer efficacement la complexité des projets logiciels, souvent marqués par l'évolution constante des exigences et des technologies. En structurant le processus autour de livrables précis à chaque étape, le SDLC présente de nombreux avantages :

- Assurer une meilleure visibilité du processus de développement pour toutes les parties prenantes ;
- Facilite l'estimation et la planification des ressources ;
- Amélioration de la maîtrise des risques ;
- Livraison régulière des logiciels et meilleure satisfaction des clients (Amazon Web Services, s.d.).

2.3. Les étapes du cycle de vie du développement logiciel

Le cycle de de du développement logiciel comprend six étapes d'après (Amazon Web Services, s.d.) :

- **Planification**

Cette étape consiste à définir les besoins du client, estimer les coûts de réalisation et planifier les ressources nécessaires. Elle permet de formaliser les objectifs à atteindre et d'élaborer un document de spécifications détaillant les attentes des parties prenantes.

- **Conception**

Apres la planification les ingénieurs choisissent ici les solutions techniques adaptées pour répondre aux exigences du client. Cela inclut la sélection des outils, l'architecture logicielle et l'intégration éventuelle de modules existants, tout en tenant compte de l'infrastructure déjà existante.

- **Implémentation**

Cette phase est dédiée au développement concret du logiciel. Les développeurs traduisent les spécifications fonctionnelles en code source, généralement en divisant le travail en tâches plus petites pour faciliter l'exécution et éviter la surcharge.

- **Test**

L'objectif est de vérifier que le logiciel fonctionne correctement et répond aux exigences définies. Cela passe par des tests automatisés et manuels afin d'identifier les beugues et d'assurer la qualité globale du produit.

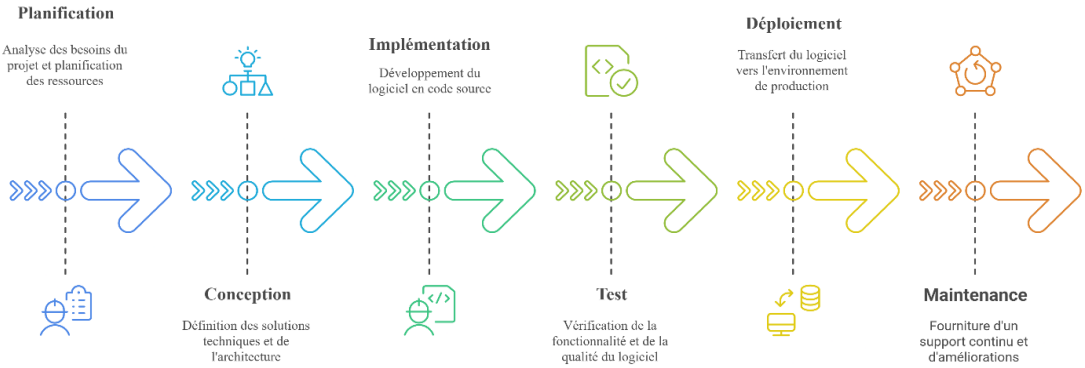
- **Déploiement**

Le logiciel est transféré de l'environnement de test vers l'environnement où il sera exploité. Cette phase comprend la préparation des packages d'installation, la configuration des serveurs et le lancement opérationnel de la solution pour les utilisateurs finaux.

- **Maintenance**

Une fois le logiciel déployé, l'équipe reste mobilisée pour corriger d'éventuelles anomalies, répondre aux retours des utilisateurs, assurer la sécurité du système et améliorer ses performances de façon continue.

Figure 1 : Les étapes du cycle de vie du développement logiciel



Source : Elaboration personnelle à partir de (Amazon Web Services, s.d.)

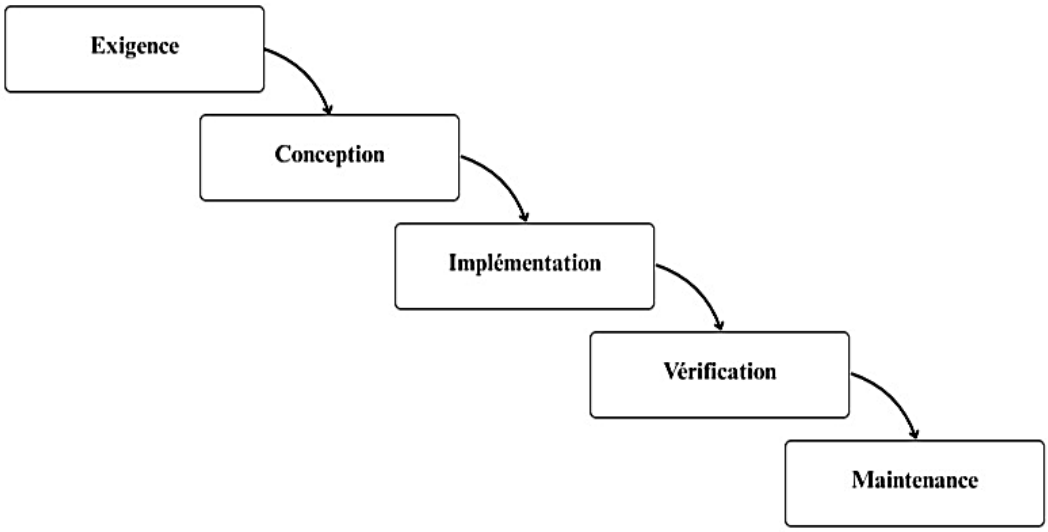
2.4. Définition des méthodes classiques de développement logiciel

Les méthodes classiques de développement logiciel également appelées modèles séquentiels, ont longtemps constitué la norme dans la gestion de projets informatiques, en particulier dans les années 1970 et 1980. Elles reposent sur une structuration linéaire et rigide du processus de développement, où les différentes phases (analyse, conception, développement, tests, déploiement) s'enchaînent de manière successive et non itérative (Tremblay, 2007).

2.4.1. Le modèle en cascade

Introduit par Royce en 1970, constitue le fondement historique de cette approche. Il se caractérise par une progression descendante, chaque étape devant être validée avant de passer à la suivante. L'un de ses apports majeurs réside dans l'intégration de la documentation à toutes les phases et dans l'attention portée à la vérification des livrables à chaque étape (Ghezzi, Jazayeri, & Mandrioli, 1991) ; (Center for Technology in Government, 1998) cités par (Tremblay, 2007). Toutefois, sa rigidité a été largement critiquée, notamment en raison de la difficulté d'anticiper précisément les besoins du client dès les premières phases du projet, ce qui augmente le risque d'inadéquation entre les spécifications initiales et les attentes réelles des utilisateurs.

Figure 2 : Le modèle cascade



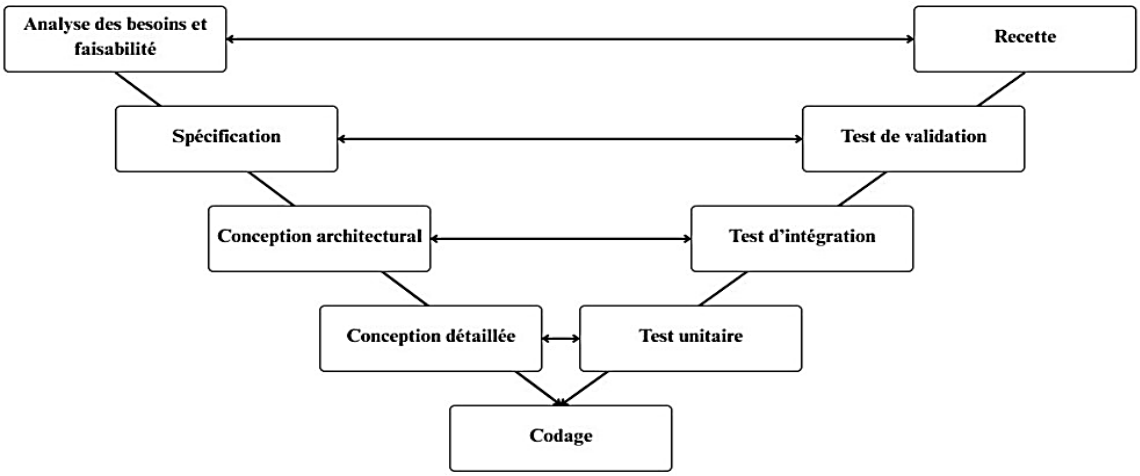
Source : Tremblay, 2007

2.4.2. Le cycle en V

Le modèle en V est une évolution qui a été conçue pour répondre aux limites du modèle en cascade. Il conserve une structure linéaire, mais introduit une correspondance explicite entre chaque phase de spécification et sa phase de validation associée, formant ainsi une structure en "V" (Tremblay, 2007). Ce modèle renforce l'importance des tests et des validations en amont du développement, et permet de mieux anticiper les exigences de qualité et de conformité.

Dans un cadre contractuel, le cycle en V est souvent valorisé pour sa capacité à sécuriser les engagements entre le client et le fournisseur, grâce à un cahier des charges rigide et détaillé en amont. Néanmoins, cette rigidité freine toute adaptation en cours de projet sauf à renégocier les termes du contrat par des avenants (Plan, 2024).

Figure 3 : Le cycle en V



Source : Tremblay, 2007

Pour conclure, les méthodes classiques offrent un cadre bien structuré, mais souffrent d'un manque de flexibilité, ce qui les rend peu adaptées à des environnements évolutifs ou incertains.

2.5. Enjeux et limites des approches traditionnelles

Malgré leur structuration rigoureuse, les méthodes classiques de développement logiciel (le modèle en cascade et le cycle en V) présentent plusieurs limites qui ont conduit à l'émergence de démarches plus souples et adaptatives, comme les méthodes agiles.

L'une des critiques majeures porte sur la rigidité du processus. Ces modèles imposent un enchaînement linéaire des étapes du projet, ce qui empêche toute révision des décisions prises dans les phases amont sans remettre en cause l'ensemble du processus (Tremblay, 2007). En pratique, cette linéarité ne correspond que rarement à la réalité des projets, où les retours en arrière et les ajustements sont fréquents.

De plus, les spécifications initiales, souvent élaborées très tôt dans le projet, sont difficilement anticipables dans leur totalité. Les utilisateurs finaux peinent à se projeter dans un système encore abstrait, ce qui rend les exigences souvent incomplètes ou imprécises (Ghezzi, Jazayeri, & Mandrioli, 1991) cité par (Tremblay, 2007). Cette lacune augmente le

risque de produire un logiciel conforme au cahier des charges mais inadéquat aux besoins réels.

Par ailleurs, le temps de latence entre la phase de conception et la livraison finale peut générer un décalage important entre les attentes initiales du client et le produit livré. Ce phénomène est accentué par l'absence d'interactions fréquentes avec le client pendant le développement (Plan, 2024). Il devient alors difficile d'intégrer des évolutions en cours de projet sans avoir recours à des modifications complexes et coûteuses.

Enfin, les modèles classiques sont peu adaptés aux environnements complexes, incertains ou en évolution rapide, comme ceux des systèmes d'information modernes. Leur manque de réactivité et d'agilité constitue un frein à l'innovation et à la satisfaction des parties prenantes.

Ainsi, bien qu'elles aient constitué une base méthodologique solide, les méthodes classiques montrent leurs limites face aux exigences contemporaines de flexibilité, d'adaptabilité et de collaboration continue.

2.6. Définition du terme « Agile »

Dans le domaine du génie logiciel, le terme Agile désigne une approche méthodologique fondée sur la capacité d'adaptation rapide et continue face à un environnement instable ou en constante évolution. Il ne s'agit pas simplement d'une métaphore littéraire de souplesse ou de légèreté, mais d'un paradigme centré sur la réactivité, l'adaptabilité et l'itération. L'agilité implique une remise en question des modèles classiques de gestion de projet linéaires, au profit de démarches plus flexibles, collaboratives et orientées vers la satisfaction continue des parties prenantes (Tremblay, 2007).

Dans cette optique, (Highsmith, 2002) définit l'agilité comme étant « la capacité conjointe de créer et de répondre aux changements dans un environnement turbulent, au bénéfice de l'entreprise ». Cette définition souligne que l'agilité ne se limite pas à une attitude réactive : elle repose également sur une anticipation proactive des transformations susceptibles d'affecter les spécifications, les technologies, les rôles des intervenants ou encore les procédures organisationnelles.

Le terme « Agile », bien qu'il soit largement diffusé et adopté dans divers secteurs, ne bénéficie d'aucune définition protégée ni standardisée, ce qui en facilite l'appropriation mais

en rend parfois l'usage imprécis. Ainsi, l'agilité doit être interprétée à la lumière du contexte dans lequel elle est mobilisée, en particulier dans le champ du développement logiciel, où elle vise à répondre aux limites des méthodes classiques en favorisant l'adaptabilité, la collaboration continue et la livraison incrémentale de valeur (Tremblay, 2007).

2.7. Origine et principes fondateurs (Manifeste agile)

Face aux limites structurelles des méthodes traditionnelles, souvent perçues comme rigides et inadaptées à l'évolution rapide des technologies, un besoin de renouvellement des pratiques de développement logiciel s'est imposé au cours des années 1990. Dans ce contexte, plusieurs praticiens expérimentés ont proposé de nouvelles approches visant à allier efficacité et flexibilité, rompant avec la logique linéaire des modèles classiques (Tremblay, 2007).

Cette dynamique a culminé lors d'une rencontre emblématique organisée le 13 février 2001 au centre de ski de Snowbird, dans l'Utah (États-Unis), où dix-sept experts du développement logiciel se sont réunis afin de confronter leurs méthodes et de formuler une vision commune. Cette réunion donna naissance au Manifeste Agile, un texte fondateur qui constitue jusqu'à aujourd'hui le pilier fondamental du mouvement agile (Tremblay, 2007).

Le manifeste proclame quatre valeurs fondamentales :

- Les individus et leurs interactions plus que les processus et les outils ;
- Des logiciels opérationnels plus qu'une documentation exhaustive ;
- La collaboration avec les clients plus que la négociation contractuelle ;
- L'adaptation au changement plus que le suivi d'un plan (Beck, et al., 2001).

Ces valeurs sont complétées par douze principes, qui insistent notamment sur la livraison fréquente de logiciel fonctionnel, la coopération étroite avec le client, la confiance accordée aux équipes de développement, ou encore l'importance d'un rythme de développement soutenable.

2.8. Les methodes agiles principales

Les méthodes agiles représentent une famille de démarches adaptatives fondées sur les principes du Manifeste Agile. Chacune propose des pratiques spécifiques, mais partage une philosophie commune axée sur l'itération rapide, la collaboration étroite avec les parties prenantes et l'adaptabilité au changement. Parmi les méthodes les plus reconnues, on peut distinguer XP, Scrum et Crystal, qui occupent une place centrale dans la littérature agile mais aussi d'autre méthodes tel que DSDM et FDD (Tremblay, 2007).

2.8.1. Dynamic Systems Development Method (DSDM)

Développé en 1994 par un consortium d'experts utilisateurs du Rapid Application Development (RAD), DSDM est une méthode agile orientée vers le respect des délais et des budgets. Elle impose une hiérarchisation stricte des exigences (MoSCoW : Must have, Should have, Could have, Won't have) et encourage la collaboration constante avec l'utilisateur final. Sa structure relativement formelle, comparée à d'autres approches agiles, en fait une méthode appréciée dans les environnements où le contrôle est essentiel (Tremblay, 2007)

2.8.2. Crystal

La méthode Crystal a été développée par Alistair Cockburn au milieu des années 1990. C'est une famille de méthodologies agiles qui s'adapte à la taille de l'équipe et à la criticité du projet. Elle valorise les interactions humaines, la communication directe et la réduction des formalités documentaires, tout en maintenant une structure souple. Chaque variante (Crystal Clear, Crystal Orange, etc.) est choisie selon le contexte spécifique du projet offrant ainsi une approche pragmatique et personnalisée (Tremblay, 2007).

2.8.3. Scrum

Scrum est la méthode la plus connue et la plus utilisée par les équipes de développement, elle a été développée par Ken Schwaber et Jeff Sutherland au début des années 1990. C'est une méthode agile centrée sur des cycles de développement courts appelés sprints, permettant une adaptation rapide aux changements et une livraison fréquente de produits fonctionnels. Elle repose sur des rôles bien définis, dont le Product Owner, le Scrum Master et l'équipe de développement, favorisant l'auto-organisation et la transparence. Cette approche met l'accent sur la planification itérative et les réunions régulières pour ajuster les priorités selon les besoins du client (Tremblay, 2007).

2.8.4. Feature Driven Development (FDD)

La méthodes FDD a été introduit en 1997 par Jeff De Luca en collaboration avec Peter Coad dans le cadre d'un grand projet bancaire à Singapour. Elle repose sur une planification initiale détaillée suivie par des itérations courtes centrées sur des fonctionnalités mesurables. Chaque fonctionnalité est conçue pour être développée en quelques jours, ce qui permet de suivre précisément l'avancement du projet. FDD met en avant la modélisation du domaine, la revue régulière du code et une gestion par objectifs, ce qui la rend adaptée aux projets complexes nécessitant un suivi rigoureux (Tremblay, 2007).

2.8.5. Extreme Programming (XP)

Extreme Programming a été conçu par Kent Beck à la fin des années 1990. Elle se distingue par une focalisation sur la qualité du code et la satisfaction du client Tout en s'appuyant sur des pratiques techniques rigoureuses telles que le développement itératif, la programmation en binôme (pair programming), les tests automatisés et la conception simple. XP vise aussi à améliorer la flexibilité et la robustesse du produit tout en favorisant une communication continue entre les développeurs et le client (Tremblay, 2007).

Synthèse des méthodes agiles les plus connues :

Tableau 2 : Synthèse des méthodes agiles

Méthode	Créateur(s)	Année de création	Caractéristiques clés
DSDM	Consortium d'experts RAD	1994	Respect des délais et budgets, hiérarchisation des exigences (MoSCoW), collaboration constante avec l'utilisateur.
Crystal	Alistair Cockburn	1994–1995	Approche adaptable selon la taille et la criticité du projet, valorise la communication humaine et la flexibilité.
Scrum	Ken Schwaber et Jeff Sutherland	Début des années 1990 (publié en 1995)	Sprints courts, rôles définis (Product Owner, Scrum Master), auto-organisation et réajustements fréquents.
FDD (Feature Driven Dev.)	Jeff De Luca et Peter Coad	1997	Développement par fonctionnalités, planification détaillée, suivi rigoureux, modélisation du domaine.

Extreme Programming (XP)	Kent Beck	1999	Qualité du code, développement itératif, tests automatisés, programmation en binôme, forte collaboration client.

Source : Elaboration personnelle à partir de (Tremblay, 2007)

2.9. Définition de l'optimisation des processus

L'optimisation des processus désigne l'ensemble des actions qui ont pour but d'améliorer la performance des activités d'une organisation, en corrigeant les dysfonctionnements, en réduisant les pertes et en augmentant la valeur produite. Elle constitue l'un des piliers fondamentaux de l'approche processus, intervenant après la phase de modélisation, et s'appuie sur une démarche structurée incluant la cartographie des processus, la priorisation des processus clés, et la mise en place d'actions correctives qui ciblent les outputs, les points faibles, les exigences du processus et finir par la mise en place des indicateurs de performance (Toumi Amara & Kouloughli, 2021).

La mise en place d'indicateurs de performance est un des leviers essentiels de l'optimisation, cela permet de mesurer les écarts, d'objectiver les résultats, et de suivre l'évolutions des activités en cours. L'indicateur est défini comme un outil d'évaluation permettant de quantifier un phénomène, d'en observer la dynamique dans le temps, et de guider les décisions managériales (Krebs, 2004), cité par (Toumi Amara & Kouloughli, 2021)

Pour être efficace, un indicateur doit répondre à plusieurs critères : il doit être pertinent, précis, reproductible, fiable, synthétique, et partagé avec le personnel afin de favoriser son implication dans la démarche d'amélioration continue.

Ainsi, l'optimisation des processus ne se limite pas à une simple amélioration technique, mais s'inscrit dans une vision transversale de pilotage de la performance au service de la stratégie de l'entreprise.

2.10. Le modèle CMMI

Le CMMI est un cadre de référence développé par le Software Engineering Institute visant à améliorer les processus au sein des organisations, notamment dans les domaines du

développement logiciel, de l'ingénierie système et des services. Il repose sur le principe que des processus bien définis et maîtrisés permettent de produire des logiciels de meilleure qualité, dans des délais plus courts et avec un moindre coût de maintenance (Chrissis, Konrad, & Shrum, 2003) cité par (Sutherland, Jakobsen, & Johnson, 2009)

Le modèle est structuré en cinq niveaux de maturité, chacun représentant un degré croissant de formalisation, de contrôle et d'amélioration continue :

- **Niveau 1 (Initial)**

  À ce niveau, l'organisation ne dispose pas de processus standardisés. Les résultats des projets sont imprévisibles et dépendent largement des compétences individuelles des membres de l'équipe. Les projets peuvent varier considérablement en termes de qualité et de délais, ce qui entraîne souvent des échecs ou des retards.

- **Niveau 2 (Géré)**

  Au niveau géré, l'organisation commence à établir des processus définis pour la gestion des projets. Ces processus sont appliqués au niveau des projets individuels, permettant une certaine prévisibilité dans les résultats. Cependant, il peut encore y avoir des variations dans l'application des processus d'un projet à l'autre.

- **Niveau 3 (Défini)**

  À ce stade, l'organisation standardise ses processus à l'échelle organisationnelle. Les pratiques de gestion de projet sont documentées et suivies de manière cohérente à travers tous les projets. Cela permet d'améliorer la qualité des résultats et d'assurer une meilleure communication entre les équipes.

- **Niveau 4 (Quantitativement maîtrisé)**

  Au niveau quantitativement maîtrisé, l'organisation utilise des données quantitatives pour piloter ses projets. Des mesures de performance sont mises en place pour évaluer l'efficacité des processus et des résultats. Cela permet une gestion proactive des projets, avec des ajustements basés sur des analyses de données.

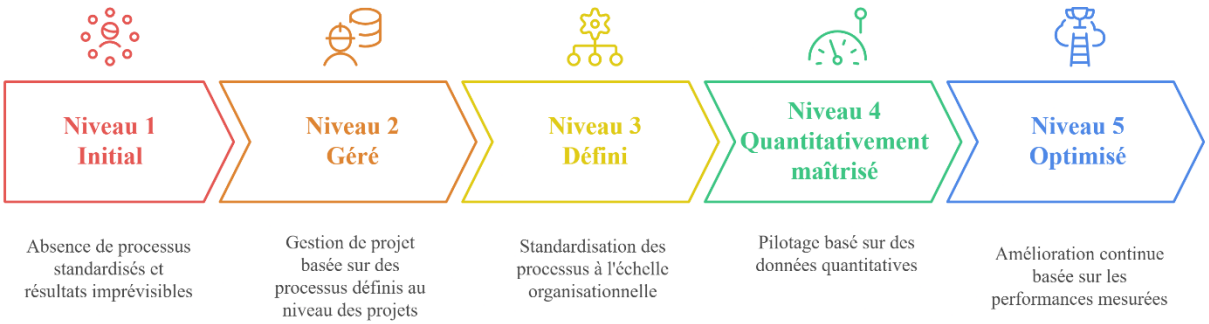
- **Niveau 5 (Optimisé)**

  Enfin, au niveau optimisé, l'organisation s'engage dans une amélioration continue de ses processus. Les performances sont régulièrement mesurées et analysées, et des initiatives

sont mises en place pour optimiser les processus en fonction des résultats obtenus. Cela permet à l'organisation de s'adapter rapidement aux changements et d'améliorer constamment sa gestion de projet.

L'intérêt majeur de CMMI réside dans sa capacité à structurer l'organisation autour de processus robustes et reproductibles. Il facilite la capitalisation des bonnes pratiques et le contrôle de la qualité à grande échelle. Cependant, ce cadre peut parfois être perçu comme rigide et peu adaptable, notamment dans des environnements changeants ou à forte innovation (Sutherland, Jakobsen, & Johnson, 2009).

Figure 4 : Les niveau du modèle CMMI



Source : Elaboration personnelle à partir de (Sutherland, Jakobsen, & Johnson, 2009)

2.11. Définition de DevOps, principes et utilité

Le terme DevOps est la contraction des mots Development (qui se traduit en français par développement) et Operations (qui se traduit en français par exploitation). Il désigne à la fois une culture, une philosophie et une stratégie opérationnelle visant à améliorer la collaboration entre les équipes de développement et celles d'exploitation, dans le but de livrer plus rapidement et de manière plus fiable des applications ou des services numériques (Microsoft Corporation, s.d.)

Pour la bonne intégration de DevOps des principes fondamentaux ont été mis en place :

- L'intégration continue (CI) : les modifications du code sont automatiquement testées et intégrées fréquemment, afin de détecter les erreurs rapidement.
- La livraison continue (CD) : les applications sont préparées à être mises en production à tout moment grâce à des processus automatisés.

- Le déploiement continu : les mises en production peuvent être réalisées automatiquement dès qu'une version est validée.
- La collaboration interdisciplinaire : les développeurs et les opérationnels travaillent ensemble dès les premières phases du projet.
- L'automatisation des tests et de la configuration : pour renforcer la fiabilité et accélérer les livraisons.
- L'amélioration continue : les performances du processus sont évaluées en permanence pour favoriser l'innovation et la réactivité (Gravier, 2018).

L'adoption de DevOps présente des avantages significatifs pour les organisations confrontées aux exigences de rapidité, de qualité et d'adaptabilité des services numériques. Cette approche permet d'aligner plus efficacement les objectifs des équipes de développement (Dev) et d'exploitation (Ops), en instaurant une culture de collaboration continue et en s'appuyant sur des pratiques d'automatisation.

Tout d'abord, DevOps permet une réduction notable du time-to-market, c'est-à-dire le temps nécessaire pour qu'une fonctionnalité passe du stade de l'idée au stade de mise en production. Grâce à l'automatisation des phases de test, de validation et de déploiement, les équipes peuvent livrer plus fréquemment des versions stables du logiciel, ce qui accroît la réactivité de l'entreprise face aux changements du marché (Gravier, 2018) et (Microsoft Corporation, s.d.)

Ensuite, la qualité des livrables est aussi améliorée. Les pratiques telles que l'intégration continue (CI) et la livraison continue (CD) permettent une détection précoce des erreurs, réduisant ainsi les coûts liés à la correction de bugs en fin de cycle. L'automatisation des tests garantit une meilleure couverture fonctionnelle, et la gestion des configurations limite les incohérences entre les environnements de développement, de test et de production (Gravier, 2018).

La fiabilité des déploiements est également un bénéfice à ne pas négliger. DevOps standardise et automatise les processus de mise en production, ce qui réduit fortement le risque d'erreurs humaines. Des mécanismes de retour arrière automatique peuvent être mis en place, renforçant ainsi la résilience du système d'information (Gravier, 2018).

Sur le plan organisationnel, DevOps contribue à réduire les coûts opérationnels. La simplification des processus, la diminution des interventions manuelles et une meilleure

coordination des équipes permettant de toujours maintenir et faire évoluer les applications (Microsoft Corporation, s.d.).

Enfin, DevOps favorise une meilleure entente entre les équipes, en rompant avec les silos traditionnels entre développement et exploitation. Cette dynamique collaborative permet de mieux comprendre les contraintes et enjeux respectifs, diminuant ainsi les tensions et favorisant une culture d'amélioration continue orientée vers la performance collective (Gravier, 2018).

2.12. Le référentiel ITIL (Information Technology Infrastructure Library)

Information Technology Infrastructure Library (ITIL) est un référentiel de bonnes pratiques destiné à la gestion des services informatiques, il propose un ensemble structuré de processus et de recommandations visant à améliorer l'efficacité, la qualité et la continuité des services offerts par les systèmes d'information. ITIL repose sur une approche orientée services et met l'accent sur la création de valeur pour les clients et les utilisateurs.

Le cadre ITIL fonctionne selon une logique de cycle de vie des services introduite dans la version V3, découpé en cinq grandes étapes interdépendantes, chaque étape comprend des processus spécifiques permettant de planifier, concevoir, délivrer et améliorer les services informatiques :

- **La stratégie des services (Service Strategy)**

  Cette première phase définit les orientations globales. L'objectif est de comprendre les besoins des clients, d'analyser la valeur attendue et de concevoir une offre de services cohérente avec les objectifs métiers. Elle aide à prendre des décisions sur les investissements, les priorités et le portefeuille de services.

- **La conception des services (Service Design)**

  Ici, les services sont pensés dans le détail : architecture, processus, ressources humaines et techniques. Cette phase vise à s'assurer que les nouveaux services sont fiables, efficaces et conformes aux exigences. Elle prévoit aussi les plans de continuité, de sécurité et de gestion des niveaux de service.

- **La transition des services (Service Transition)**

Cette étape fait office de pont entre la phase de conception et la phase d'exploitation. Elle comprend des processus tels que la gestion du changement, des configurations et des mises en production. Le but est d'introduire les services dans l'environnement opérationnel avec un minimum de risques et de perturbations.

- **L'exploitation des services (Service Operation)**

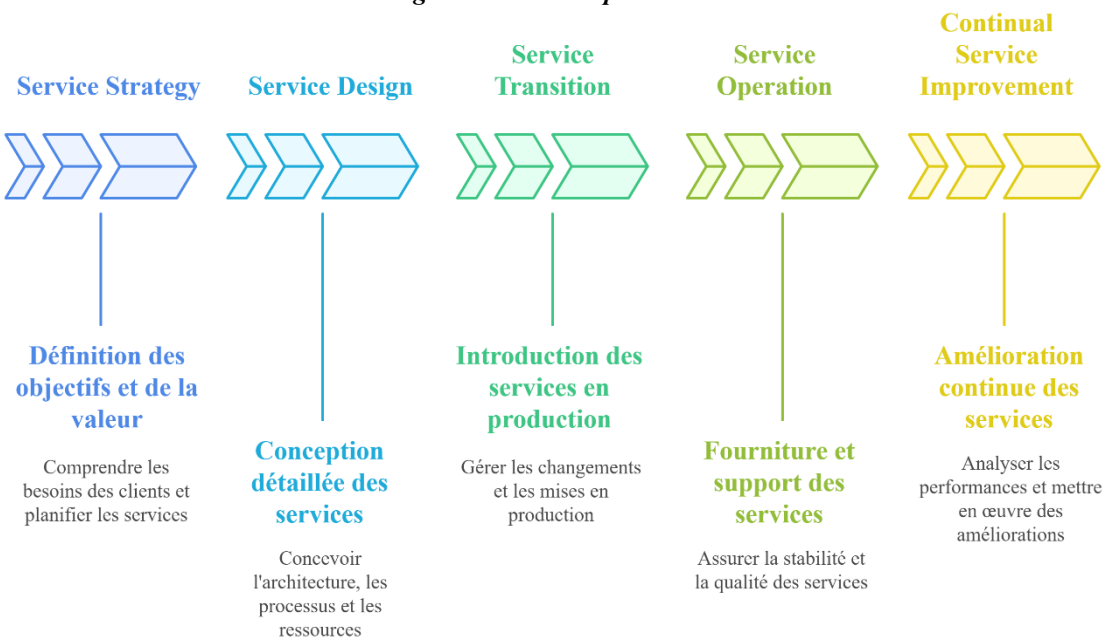
Dans cette phase les services sont délivrés aux utilisateurs finaux. Elle englobe la gestion des incidents, des accès et des événements. L'objectif est d'assurer une fourniture continue et stable des services avec un niveau de qualité conforme aux attentes.

- **L'amélioration continue des services (Continual Service Improvement)**

Cette phase finale vise à identifier et mettre en œuvre des améliorations permanentes, fondées sur des indicateurs de performance. Elle permet de tirer des enseignements des expériences passées pour faire évoluer les services et les processus de gestion.

L'un des principaux avantages d'ITIL est la possibilité d'améliorer en continu la performance du système d'information et ça grâce à son approche structurée et adaptable, il permet aussi de professionnaliser la gestion des services informatiques et d'augmenter leurs fiabilités tout en maîtrisant les coûts et les incidents. ITIL facilite aussi la prise de décision stratégique en fournissant des indicateurs de performance pertinents. (Chakir, Medromi, & Sayouti, 2012).

Figure 5 : Les 5 étapes de ITIL



Source : Elaboration personnelle à partir de (Chakir, Medromi, & Sayouti, 2012)

CHAPITRE II : CADRE METHODOLOGIQUE ET ORGANISATIONNEL

Section 1 : Présentation de l'organisme d'accueil

Cette section vise à fournir un aperçu général de l'organisme dans lequel le stage a été réalisé. Elle présente sa structure et ses missions principales.

1.1. Présentation de l'entreprise Naftal

Naftal est une société algérienne par actions (SPA), filiale du groupe SONATRACH, spécialisée dans la distribution et la commercialisation des produits pétroliers et dérivés sur le marché national (Naftal, s.d.)

Créée initialement sous le nom d'Entreprise de Raffinage et de Distribution des Produits Pétroliers (ERDP) par le décret n° 80/101 du 6 avril 1980, elle entre en activité en juin 1982. La séparation de l'activité de raffinage en 1987 donne lieu à la restructuration de l'entreprise, qui conserve désormais uniquement les fonctions de distribution et de commercialisation. Le 18 avril 1998, elle adopte le statut de (SPA) avec un capital social de 16 milliards de dinars, tout en demeurant intégralement détenue par SONATRACH.

Sur le plan organisationnel, Naftal repose sur des divisions internes importantes, aussi bien en termes de moyens humains ou de matériels qui sont réparties sur l'ensemble du territoire algérien, ce qui garantis une couverture nationale efficace (Naftal, s.d.). En 2024, elle a commercialisé 17,1 millions de tonnes de produits pétroliers, générant un chiffre d'affaires de 434,1 milliards de dinars et employant plus de 30 000 agents.

Naftal propose une gamme étendue de produits destinée aussi bien aux particuliers qu'aux professionnels. Pour les particuliers, elle commercialise divers types de carburants (essence sans plomb, essence normale, super, gasoil, GPL/C), des GPL domestiques (butane, propane, Sirghaz), des lubrifiants, ainsi que des pneumatiques. Du côté des professionnels, elle dessert plusieurs secteurs comme la marine, l'aviation, le BTP (à travers les bitumes), ou encore les industries pharmaceutiques et cosmétiques via des produits spéciaux comme les paraffines, cires, essences et solvants (Naftal, s.d.).

Pour assurer la distribution de ses produits à travers l'ensemble du territoire national, Naftal s'appuie sur une infrastructure solide résumée dans le tableau suivant :

Dépôts carburants terre	Centres & mini-centres GPL	Centres vrac GPL	Dépôts relais	Dépôts aviation	Centres marine	Centres bitumes	Centres lubrifiants & pneumatiques	Réseau pipelines (km)	Parc roulant (unités)	Stations-service (total)	Stations en gestion directe
41	41	10	48	30	6	15	24	700	3 300	1 013	380

Tableau 3 : Les moyens de Naftal

Source : Site officiel de Naftal

Le tableau ci-dessous résume les informations essentielles à retenir concernant l'entreprise Naftal :

Nom complet	**Naftal - Société Nationale de Commercialisation et de Distribution de Produits Pétroliers**
Statut juridique	Société par actions (SPA) (100 % des actions détenues par le groupe SONATRACH)
Date de création	1982 (origine ERDP créée en 1980)
Changement de statut	Devient SPA le 18 avril 1998
Filiale de	Groupe SONATRACH
Mission principale	Distribution et commercialisation de produits pétroliers et dérivés sur le territoire national
Chiffres clés (2024)	• 17,1 millions de tonnes de produits pétroliers par an • 434,1 milliards DA de chiffre d'affaires • 32 328 employés
Implantation	Présence nationale à travers un réseau de stations-service

Produits pour particuliers	• **Carburants (sans plomb, gasoil)** • **GPL/C (Sirghaz)** • **GPL domestique (butane, propane)** • **Lubrifiants** • **Pneumatiques**
Produits pour professionnels	• Carburants et lubrifiants marines et aviation • Bitumes (modifies aux polymers, purs, oxydés, fluidifiés, émulsions) • Produits spéciaux (paraffines, cires, essences spéciales, solvants)
Secteurs bénéficiaires	Automobile, transport, BTP, industrie, aviation, marine, pharmaceutique, cosmétique, agro- alimentaires
Position sur le marché	Leader national dans la distribution de produits pétroliers

Tableau 4 : Les informations essentielles sur Naftal

Source : Elaboration personnelle à partir du site officiel de Naftal

1.2. Organisation de la société

Comme mentionné précédemment, Naftal regroupe un nombre important de ressources, tant humaines que matérielles, qu'elle organise de manière structurée afin d'assurer l'efficacité de ses missions. L'entreprise a jugé qu'il est plus judicieux de répartir ces ressources selon un modèle ordonné, en établissant des relations d'autorité entre les différentes entités. Ce modèle permet de diviser les objectifs stratégiques de l'entreprise en tâches spécifiques, regroupées au sein de services, départements et directions, chacun placé sous la responsabilité d'un manager disposant de l'autorité et privilèges nécessaires.

L'organisation de Naftal repose sur deux niveaux complémentaires :

- Les structures centrales sont chargées de la définition des orientations stratégiques et du suivi global des activités.
- Les structures opérationnelles sont responsables de l'exécution sur le terrain. Ces dernières assurent la distribution des produits commercialisés par l'entreprise sur l'ensemble du territoire national.

Enfin, l'organigramme de Naftal reflète cette structuration hiérarchique. Il s'articule autour de trois grandes composantes, incluant une direction générale appuyée par un staff composé d'un Comité Exécutif, d'un Comité de direction, de conseillers, ainsi qu'une direction projets, permettant une coordination efficace des différentes unités.

Les Structures Opérationnelles sont composées de trois (03) Branches :

- Branche Carburant.
- Branche Commercialisation.
- Branche GPL.

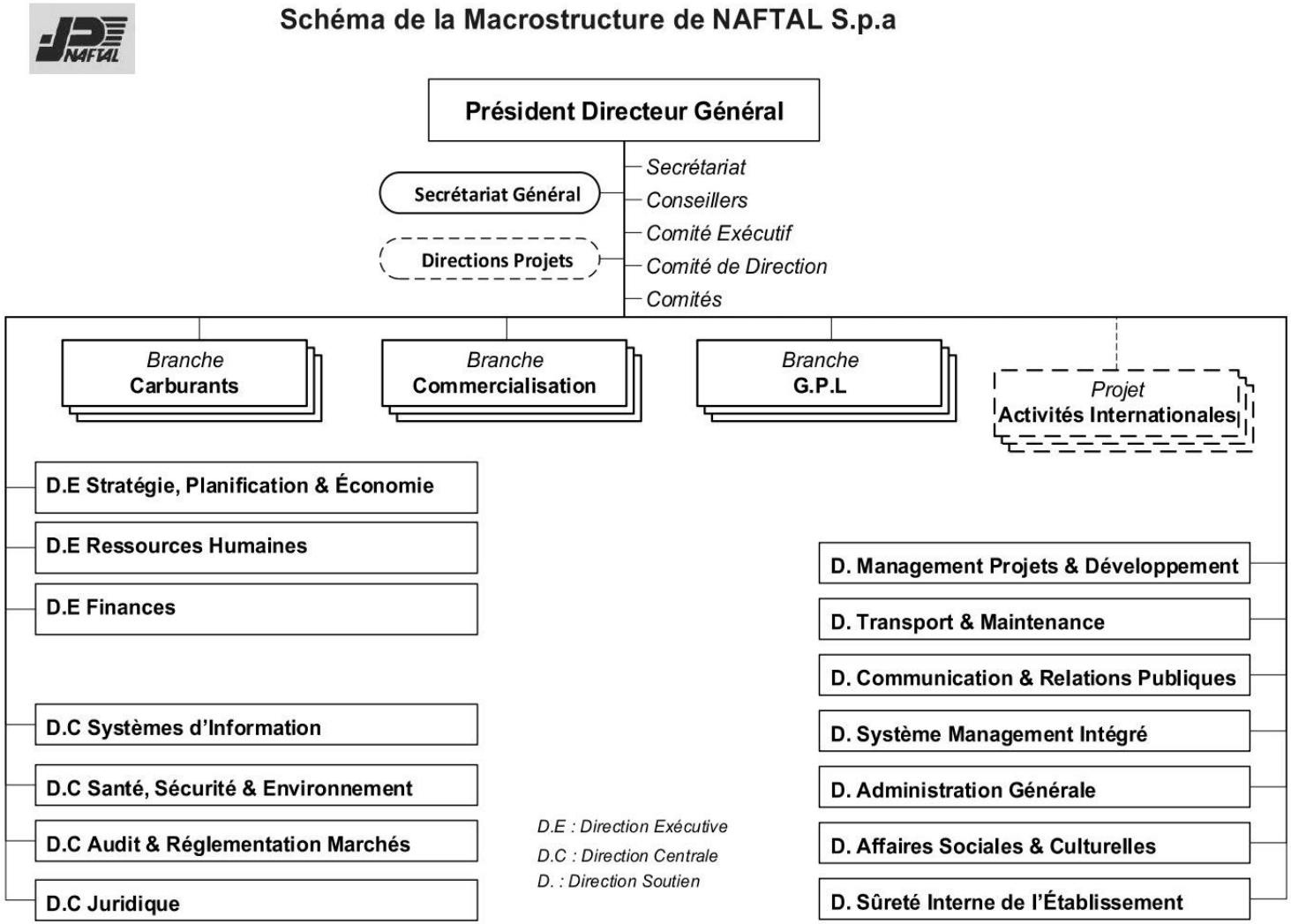
Les structures fonctionnelles qui sont composées de trois Directions (03):

- Directions exécutives qui sont au nombre de trois (03) :
  - Direction Exécutive Stratégies, Planification & Économie.
  - Direction Exécutive Finance.
  - Direction Exécutive Ressources Humaines.
- Directions centrales qui sont au nombre de quatre (04) :
  - Direction centrale des systèmes d'information.
  - Direction centrale de santé, sécurité et environnement.
  - Direction centrale d'audit et de réglementation marchés.
  - Direction centrale juridique.
- Directions de Soutiens qui sont :
  - Direction management projets et développement.

- Direction de transport et maintenance.
- Direction de la communication et des relations publiques.
- Direction du système management intégré.
- Direction d’administration générale.
- Direction des affaires sociales et culturelles.
- Direction de sureté interne de l’établissement.

L'organigramme de l'entreprise se présente comme ceci :

Figure 6 : L'organigramme de l'entreprise Naftal



Source : Document interne de Naftal

1.3. Présentation de la direction centrale des systèms d'information (DCSI)

La Direction Centrale des Systèmes d'Information (DCSI) constitue un pilier stratégique de la transformation numérique chez Naftal, elle est structurée en quatre grandes directions composées de départements spécialisés, elle est chargée de moderniser l'ensemble du système d'information afin de répondre aux exigences opérationnelles, techniques, sécuritaires et documentaires de l'organisation. Ses missions s'étendent de la migration des anciens systèmes vers des plateformes plus performantes à l'intégration de logiciels et solutions modernes, en passant par, la sécurisation des données, la gouvernance documentaire et la mise en place d'outils d'aide à la décision. À travers une approche coordonnée la DCSI veille à la cohérence technologique, à la performance des systèmes et à l'alignement des projets informatiques avec les besoins stratégiques de la société.

La DCSI se compose de quatre directions (04):

- Direction infrastructures

Cette direction est chargée de définir et mettre en œuvre l'architecture des systèmes, des bases de données et réseaux d'infrastructure du Système d'Information.

- Direction solutions métiers

Son rôle est de concevoir et réaliser des solutions informatiques qui répondent aux besoins opérationnels de l'ensemble des Structures de la Société.

- Direction opérations

Elle est chargée de veiller au bon fonctionnement des plateformes monétiques et décisionnelles en plus de garantir la disponibilité du matériel informatique dédié aux utilisateurs finaux du système d'information de la société.

- Direction sécurité & conformité

Le but de cette direction est de concevoir et mettre en place un dispositif permettant la sécurité et la pérennité des systèmes d'information mis en place.

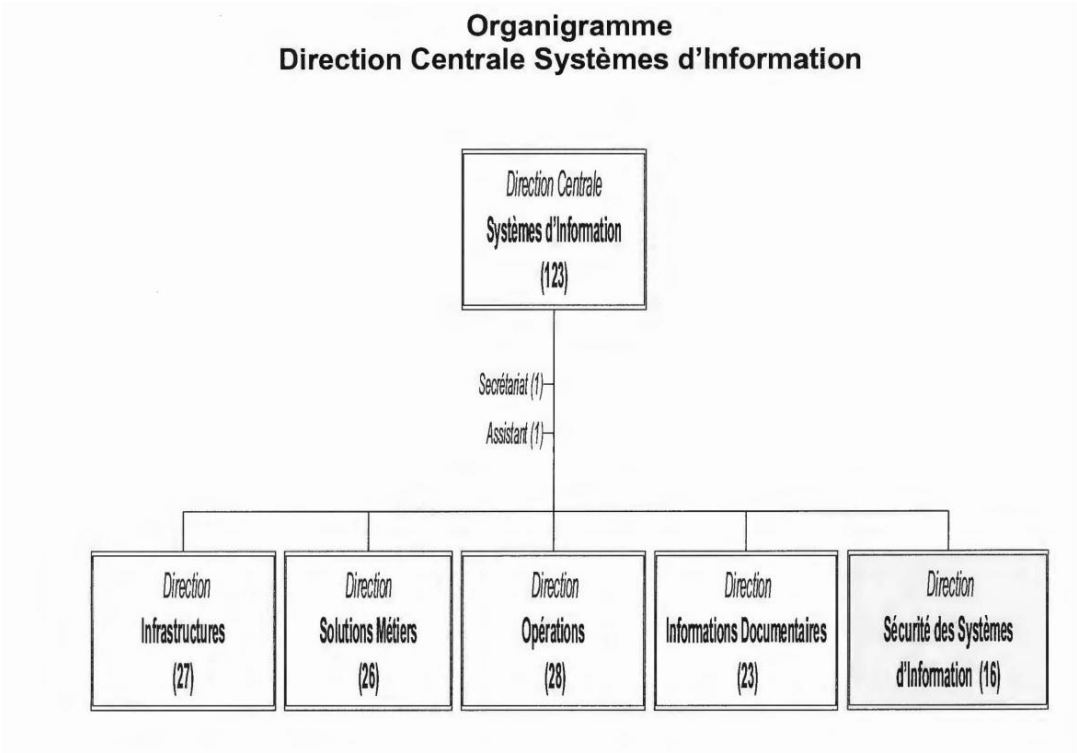
- Direction informations documentaires

Sa mission principale est de recueillir et analyser les besoins de versement et d'élimination des documents ainsi que des acquisitions (ouvrages, revues, etc.) et identifier les sites à inspecter, mais elle est aussi chargée d'élaborer :

- Un projet programme annuel de versement, d'éliminations et d'archivage électronique des documents,
- Un projet programme annuel d'acquisitions,
- Un projet programme annuel des missions d'inspection,
- Suivi des activités archives de ta société.

L'organigramme de la DCSI chez Naftal se présente comme ceci :

Figure 7 : L'organigramme de la DCSI



Source : Document interne de Naftal

Section 2 : Cadre méthodologique

Cette section décrit la démarche adoptée pour mener à bien l'étude. Elle précise les choix méthodologiques effectués ainsi que les outils utilisés.

2.1. La méthodologie

La méthodologie, dans le champ des sciences sociales et de gestion, est définie comme étant l'usage approprié des méthodes de recherche pour aboutir à un résultat souhaité.

Selon (Aktouf, 1987), elle se définit comme l'étude du bon usage des méthodes et techniques, en mettant l'accent sur la nécessité de les adapter rigoureusement à l'objet de la recherche ainsi qu'aux objectifs poursuivis.

D'un autre côté, (Gauthier, 2009) cité par (Melloud) élargit sa définition de la méthodologie de recherche qui va englober à la fois l'organisation de la pensée et la mise en œuvre pratique à travers des méthodes spécifiques.

Enfin, (Gavard-Perret, Gotteland, Haon, & Jolibert, 2012) établit un lien entre la méthodologie et l'épistémologie en la définissant comme l'étude des méthodes utilisées pour produire des connaissances et ceci tout en précisant que l'épistémologie ne se limite pas uniquement à la méthodologie.

À titre comparative, (Aktouf, 1987) insiste principalement sur l'adaptation rigoureuse des outils de recherche aux contextes particuliers, tandis que (Gauthier, 2009) cité par (Melloud) met l'accent sur l'esprit du chercheur qui va par la suite définir la manière dont les outils pratiques vont être utilisés. (Gavard-Perret, Gotteland, Haon, & Jolibert, 2012) pour sa part, replace la méthodologie dans une perspective plus théorique en lien avec la philosophie des sciences.

Ainsi, si les trois auteurs reconnaissent l'importance de la méthodologie pour produire une recherche de qualité, ils ont tout de même des visions complémentaires : pratique pour Aktouf, organisationnelle pour Gautier, et épistémologique pour Avenier.

2.2. La recherche qualitative

La recherche qualitative est une approche qui vise à comprendre en profondeur les phénomènes sociaux, organisationnels ou humains en s'intéressant aux représentations, aux

pratiques et aux interactions des acteurs. Contrairement à la recherche quantitative, elle ne cherche pas à mesurer ou à généraliser les résultats à une population plus large, mais à explorer la complexité d'une situation particulière.

Parmi ses qualités, la recherche qualitative permet d'accéder à des données détaillées, de saisir les dynamiques internes et d'analyser les processus dans leur contexte réel. Elle est particulièrement adaptée aux environnements instables et peu formalisés. Toutefois, cette approche n'est pas parfaite car elle présente aussi des limites, elle expose le chercheur à une possible subjectivité dans l'analyse des données, peut rendre difficile la reproductibilité de l'étude, et nécessite un investissement important en temps pour la collecte et l'interprétation des informations.

Dans le cadre de ce mémoire, la recherche qualitative s'est imposée comme la démarche la plus adaptée étant donné que l'objectif est de comprendre le fonctionnement réel du processus de développement logiciel, d'explorer les pratiques existantes, ainsi que les perceptions et attentes des développeurs et autres acteurs impliqués. Dans un contexte d'étude qui est caractérisé par l'absence de processus formalisées et d'outils de gestion de processus, l'observation directe ainsi que les entretiens deviennent essentielles pour saisir les réalités du terrain.

La souplesse de la recherche qualitative permet ainsi de tenir compte des spécificités du cas étudié et de proposer des recommandations d'amélioration adaptées à ses contraintes et à ses dynamiques internes.

2.3. Les sources d'informations

Afin de répondre aux objectifs de ce mémoire, plusieurs sources d'informations ont été mobilisées :

2.3.1. Le recueil documentaire

Le recueil documentaire permet de collecter des informations pertinentes à partir de diverses sources qu'elles soient internes ou externes.

- Documents internes : tels que les rapports d'activité, les organigrammes, ou encore les procès-verbaux de réunion.

- Documents externes : comme des articles académiques, des livres spécialisés, des sites Internet…etc.

Ce processus permet de bâtir une compréhension approfondie du contexte, d'identifier des lacunes dans la documentation existante, et de situer la recherche dans le cadre des connaissances préalablement établies.

2.3.2. Entretien (interview)

L'entretien ou entrevue, également appelée interview, est une méthode de collecte de données fondée sur une interaction verbale entre un intervieweur et un répondant. Elle permet d'obtenir des informations ciblées sur un sujet précis en lien avec les objectifs de la recherche. L'entretien se distingue par sa souplesse, offrant la possibilité d'adapter le degré de liberté accordé au répondant et la profondeur des informations recherchées, selon le type d'entretien choisi : directif, semi-directif ou non directif. Cette méthode est particulièrement adaptée aux recherches qualitatives visant à explorer en détail les représentations, les pratiques ou les discours des acteurs étudiés (Aktouf, 1987).

Pour cette étude deux types d'entretiens ont été employés :

- Les entretiens non directifs

Deux entretiens non directifs ont été réalisés au début de recherche. L'objectif principal de ces entretiens était de comprendre le fonctionnement général de l'entreprise, son organisation interne, ainsi que son rôle, sans oublier la structure et les services proposés par la Direction Centrale des Systèmes d'Information (DCSI). Ce type d'entretien, laissant une grande liberté d'expression au répondant, a permis de recueillir des informations riches et spontanées qui sont indispensables pour obtenir une vision globale du contexte organisationnel et orienter efficacement la suite de la démarche. Chaque entretien a duré un peu plus d'une heure, offrant ainsi le temps nécessaire pour explorer en profondeur les thématiques abordées.

- Les entretiens semi-directifs

Sept entretiens semi-directifs ont été menés auprès de chefs de projet, directeurs et développeurs appartenant à différentes équipes fonctionnelles de l'entreprise. Ces entretiens avaient pour objectif d'explorer en profondeur les pratiques actuelles de développement logiciel, d'établir un état des lieux précis et de recueillir les perceptions, les difficultés

rencontrées ainsi que les suggestions d'amélioration formulées par les acteurs directement impliqués. Guidés par une grille thématique couvrant des aspects tels que l'organisation du travail, la coordination entre les équipes et l'utilisation des outils, les entretiens ont conservé une certaine souplesse afin de laisser émerger des éléments inattendus et des discours personnels.

Cette méthode a été choisie car elle permet de structurer l'échange autour de thématique abordée tout en offrant aux répondants la liberté d'exprimer librement leur point de vue. La durée des entretiens a varié entre 50 et 70 minutes ce qui a favorisé l'approfondissement des sujets tout en maintenant un cadre suffisamment ouvert pour faire apparaître des éléments inattendus, essentiels à la compréhension du processus à optimiser.

- Guide d'entretiens

Afin d'assurer une certaine cohérence dans la conduite des entretiens semi-directifs, un guide thématique a été élaboré à la suite des deux entretiens non directifs exploratoires dans le but de répondre aux objectifs de la recherche, à savoir dresser un état des lieux détaillé des pratiques actuelles et identifier les leviers potentiels d'optimisation.

Structuré en six grandes rubriques, il couvre les dimensions essentielles à l'analyse du processus de développement logiciel : organisation générale, méthodologies utilisées, outils et technologies, pratiques de qualité, gestion des délais, ainsi que les axes d'amélioration envisagés. Ce guide a permis de cadrer les échanges tout en laissant une marge de flexibilité pour approfondir certains points en fonction des spécificités des répondants.

Le tableau ci-dessous présente les objectifs visés et l'intérêt des données collectées pour chacune des six rubriques abordées dans le guide d'entretien thématique :

Tableau 5 : Résumé du guide d'entretien

Thèmes abordés	Objectifs visés
Informations générales	Identifier le profil du répondant (fonction, service, expérience) pour contextualiser les réponses.
Organisation et méthodologies	Comprendre le déroulement du processus de développement logiciel et les méthodes appliquées.

Outils et technologies	Identifier les outils utilisés pour le développement et la gestion de projet et évaluer leur pertinence.
Qualité, tests et maintenance	Analyser les pratiques de tests, de gestion de la qualité et de suivi des incidents.
Délais et performance	Évaluer la manière dont les délais sont estimés et les indicateurs de performance utilisés.
Problèmes et opportunités	Recueillir les difficultés perçues et les pistes d'amélioration proposées par les les acteurs impliqués.
Perspectives et attentes	Explorer la vision des répondants sur l'évolution future du processus et les changements souhaités.

Source : Elaboration personnelle à partir du guide d'entretien

Note : L'ensemble des questions du guide d'entretien sont présentés dans l'Annexe A

- Les interviewés

Tableau 6 : Liste des interviewés

Nombre d'interviewés	Lieux de l'interview	Durée de l'interview	Type d'interview	Fonction de l'nterviewé(s)	Année d'experience de l'nterviewé(s)
1	Bureau	60 min	Entretien non directifs	Assistante du directeur des systèmes d'information	11 ans
1	Bureau	50 min	Entretien non directifs	Chef de projet développement	12 ans
1	Bureau	60 min	Entretien semi directifs	Directeur des opérations	23 ans
2	Bureau	50 min	Entretien semi directifs	Chefs de projet informatique	5 ans
1	Bureau	55 min	Entretien semi directifs	Directeur solution métier	22 ans
2	Bureau	60 min	Entretien semi directifs	Chefs de projet développement	4 ans
1	Bureau	70 min	Entretien semi directifs	Chef de projet développement	12 ans
1	Bureau	50 min	Entretien semi directifs	Chef département architecture et urbanisation	15 ans

1	Bureau	60 min	Entretien semi directifs	Chef département decisionnel et monitoring	20 ans

Source : Elaboration personnelle

Note : Le chef de projet développement avec 12 ans d'expérience a été interviewé deux fois : une première fois sans guide d'entretien et une seconde fois avec le guide d'entretien

2.4. Traitement et analyse des données

Comme dans tout processus de recherche l'analyse et le traitement des données recueillies constituent une étape cruciale pour garantir des résultats fiables. Cette étape vise non seulement à assurer la pertinence de la recherche, mais aussi à expliquer et structurer l'ensemble des informations collectées afin d'éclaircir les idées et de mieux comprendre le raisonnement des différents individus. Cela permet ensuite d'engager une discussion pertinente et de tirer des conclusions solides.

Les données recueillies ont été traitées et analysées selon une approche thématique, permettant de dégager des catégories récurrentes à partir du contenu des entretiens semi-directifs.

Le recours à un logiciel d'analyse qualitative tel que NVivo n'a pas été nécessaire. En effet, les réponses obtenues étaient relativement homogènes et convergentes entre les différents participants. De plus, les propos recueillis étaient suffisamment explicites et structurés pour permettre une analyse manuelle rigoureuse, sans qu'un outil informatique d'aide à l'analyse ne soit nécessaire.

Cette démarche a permis d'identifier des thèmes centraux tels que les défis organisationnels rencontrés, les outils actuellement utilisés, ainsi que les suggestions d'amélioration formulées par les répondants. L'approche thématique a ainsi facilité la mise en évidence de tendances significatives, contribuant directement à la formulation de réponses aux questions de recherche.

CHAPITRE III : RÉSULTATS ET DISCUSSION

Section 1 : Analyse du processus de développement actuel logiciel chez Naftal

Dans ce chapitre nous allons présenter l'état actuel du processus de développement logiciel chez Naftal, identifier les problèmes majeurs à résoudre et par la suite exposer les solutions et attentes des différentes parties prenantes.

1.1. Description du processus de développement logiciel actuel

Le processus de développement logiciel dans la société étudiée débute soit à la demande d'un client, soit sur initiative de la DCSI qui s'inscrit dans sa démarche d'amélioration continue.

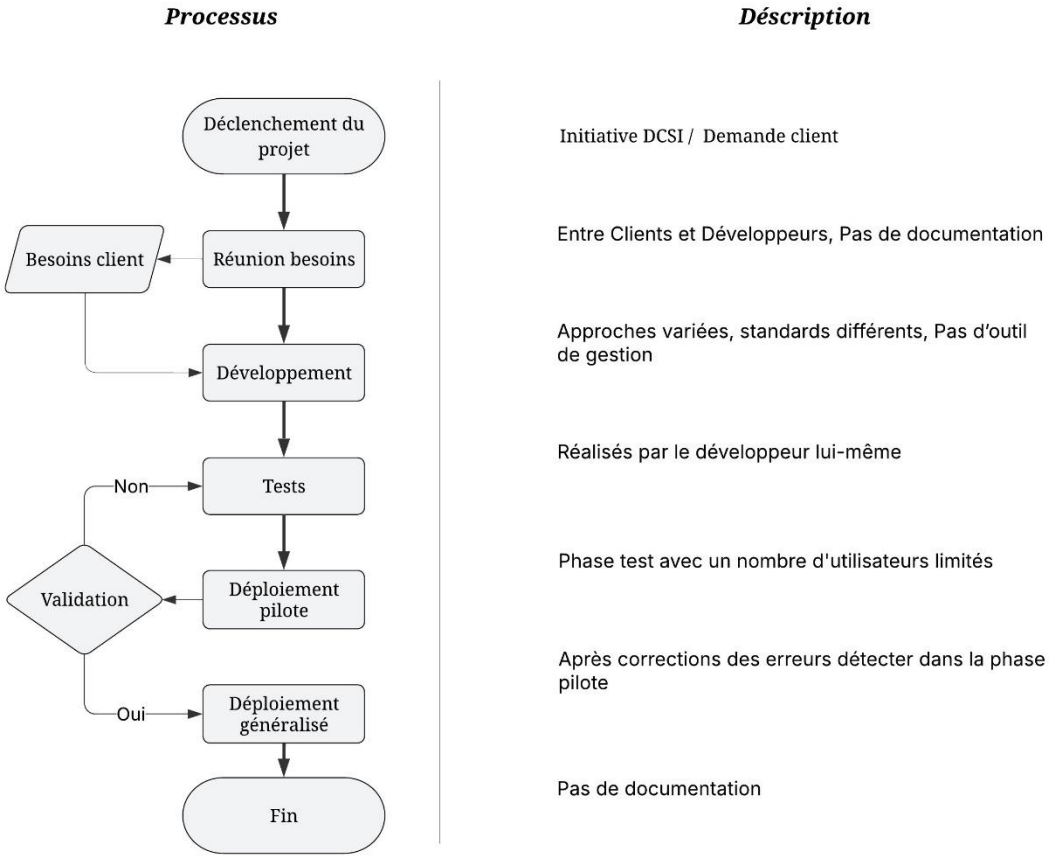
La définition des besoins se fait lors de réunions entre les clients et les développeurs, sans support documentaire structuré. Les équipes de développement sont organisées par domaine (commercial, ressources humaines, finance, etc.) et chacune applique ses propres standards de conception, ce qui peut entraîner un manque d'uniformité entre les projets.

Concernant le développement, les équipes adoptent différentes approches. Certaines travaillent collectivement sur une fonction avant de passer à la suivante, tandis que d'autres répartissent les tâches entre leurs membres, avec un responsable chargé d'assembler l'ensemble. Le test est effectué directement par le développeur ayant réalisé la fonctionnalité, ce qui peut poser des limites en matière d'objectivité et de détection d'erreurs.

Enfin, le déploiement suit une stratégie en deux phases : La première phase appelée phase pilote où la solution développée est déployée à un nombre limité d'utilisateurs ce qui permet d'identifier d'éventuels ajustements. Une fois les corrections effectuées, le déploiement est généralisé à l'ensemble des utilisateurs concernés. Cette méthode assure une certaine maîtrise des risques.

Modélisé sur la plateforme Lucidchart, le flowchart ci-dessous représente le processus actuel de développement logiciel tel qu'il a été observé dans l'entreprise. Il illustre les principales étapes, de la demande initiale au déploiement, cela va permettre d'identifier et mettre en évidence les zones de faiblesse, notamment l'absence d'outils de gestion et de documentation, qui impactent la coordination et la transmission des connaissances.

Figure 8 : Flowchart du processus de développement logiciel actuel



Source : Elaboration personnelle à partir des réponses d'entretiens

1.2. Analyse des lacune identifiées dans le processus actuel

Apres l'analyse du processus de développement logiciel actuel les lacunes majeures qui affectent l'efficacité et la performance globale des projets ont été mis en évidence.

D'une part, l'absence de documentation formalisée dès la définition des besoins représente un obstacle majeur à l'efficacité du processus de développement logiciel. En l'absence de toute trace écrite les discussions et décisions prises lors des réunions entre les clients et les développeurs ne sont pas conservées ce qui nuit à la traçabilité des besoins et à la clarté des attentes, ce manque de documentation peut engendrer des incompréhensions, des modifications fréquentes des exigences et des écarts par rapport aux attentes initiales du client. De plus, l'absence de documentation à chaque étape du processus après la définition des besoins jusqu'au déploiement, pose un problème de capitalisation des connaissances. Les décisions prises, les choix techniques et les enseignements tirés des projets précédents

risquent ainsi de se perdre en particulier en cas de départ d'un membre clé de l'équipe, réduisant la capacité à tirer parti des retours d'expérience pour améliorer les futurs projets.

D'autre part, l'absence d'outils de gestion de projet constitue une faiblesse structurelle, sans solution dédiée comme un logiciel de gestion de projet tel que Jira pour planifier, suivre et coordonner les tâches, les équipes opèrent de manière informelle, ce qui rend difficile le pilotage des délais, l'identification des priorités, le suivi de l'avancement ou même l'évaluation des performances.

Par ailleurs, chaque équipe applique ses propres standards de conception ce qui entraîne un manque d'uniformité entre les projets. Cette hétérogénéité ajoutée à l'absence de documentation complique la maintenance, le travail collaboratif entre équipes et entrave la scalabilité des solutions développées.

Enfin, le fait que les tests soient réalisés par les mêmes développeurs ayant codé les fonctionnalités limite l'objectivité du contrôle qualité. En l'absence d'une évaluation effectuée par une partie neutre certaines erreurs peuvent passer inaperçues augmentant ainsi le risque d'incidents en production.

Voici à présent un tableau qui résume l'entièreté des problèmes révélés :

Tableau 7 : Résumé des problèmes révélés

Problème détecté	Description	Conséquence
Absence d'outils de gestion de projet	Aucun outil n'est utilisé pour organiser ou suivre l'avancement.	Coordination difficile, manque de visibilité sur l'état d'avancement, risques de retards.
Absence de documentation	Aucune phase n'est documentée, ce qui	Manque de traçabilité perte de connaissances en cas de départ d'un membre de l'équipe.
Hétérogénéité des standards	Chaque équipe applique ses propres standards de conception.	Risques d'incohérences entre les projets, difficultés de maintenance.
Définition des besoins non formalisée	Les besoins sont définis oralement sans support structuré.	Risque d'ambiguïté ou d'oubli.
Tests réalisés par les développeurs eux-mêmes	Le développeur teste sa propre fonctionnalité.	Moins d'objectivité, et augmentation du risque d'erreurs non détectées.

Manque d'effectif	Les développeurs assurent toutes les étapes du processus (analyse, développement, tests, déploiement).	Surcharge de travail et des risques de qualité ou de délais.

Source : Elaboration personnelle à partir des réponses d'entretiens

Ces lacunes, prises ensemble, affaiblissent la coordination entre les parties prenantes, augmentent les risques opérationnels et freinent l'amélioration continue des processus. Elles soulignent la nécessité d'une structuration plus rigoureuse du processus appuyée par des outils adaptés, des standards communs et des pratiques collaboratives renforcées.

Section 2 : Analyse des problèmes et leviers d'optimisation

Cette section a pour objectif de classer les principaux dysfonctionnements observés dans le processus de développement logiciel actuel. Chaque problématique est ensuite comparée avec les modèles théoriques et les bonnes pratiques issues de la littérature afin de dégager des axes pertinents d'optimisation.

2.1. Classification des problèmes détectés

L'analyse détaillée du processus de développement logiciel actuel a permis de mettre en évidence les problèmes majeurs qui freinent l'efficacité des projets et la performance globale de l'organisation. Ces problèmes peuvent être regroupés en trois grandes catégories : organisationnels, techniques et humains.

- **Problèmes organisationnels**
  - Absence d'outils de gestion de projet pour planifier, suivre et coordonner les tâches.
  - Faible traçabilité des décisions, en raison de l'absence de documentation formelle.
  - Difficultés à assurer la continuité des projets en cas de départ d'un membre clé, faute de transmission organisée des connaissances.
- **Problèmes techniques**
  - Multiplicité des standards de conception entre les équipes (naming conventions, structuration des bases de données, méthodes de codage), entraînant des incohérences.

- Absence d'intégration d'outils collaboratifs permettant de centraliser les tâches, les codes sources et les informations projet.
- Tests effectués sans procédure standardisée ni vérification croisée, limitant la fiabilité et la robustesse des livrables.

- **Problèmes humains**
  - Répartition floue des rôles et responsabilités, certains développeurs devant assumer des tâches multiples allant de l'analyse des besoins au déploiement.
  - Manque de formation sur les bonnes pratiques en matière de développement collaboratif.
  - Résistance potentielle au changement, en raison d'habitudes ancrées et de pratiques informelles.

2.2. Comparaison avec les modèles de référence

Les problèmes relevés dans le processus actuel de développement logiciel peuvent être éclairés et analysés à la lumière de plusieurs modèles reconnus en ingénierie logicielle et en management des systèmes d'information.

- **Par rapport au cycle en V**

Chez Naftal, l'absence de documentation structurée, de procédures de validation formelle et de tests indépendants révèle un écart important par rapport aux bonnes pratiques proposées par ce modèle. Cela entraîne une difficulté à garantir la qualité, à remonter aux besoins initiaux et à vérifier la conformité des livrables.

- **Par rapport aux méthodes agiles**

Les approches agiles valorisent l'adaptabilité, la collaboration étroite avec le client, les itérations rapides et les feedbacks fréquents. Cependant, même dans un cadre agile, certaines conditions sont nécessaires : définition claire des rôles (Product Owner, Scrum Master, équipe de développement), transparence sur l'avancement via des outils comme les backlogs et la mise en place de rituels (daily meetings, rétrospectives).

Ici, bien que Naftal adopte une certaine flexibilité, elle ne bénéficie pas des leviers agiles essentiels : il manque une structuration des rôles, des mécanismes d'ajustement continu et des outils permettant de suivre visuellement l'avancement.

- **Par rapport au modèle CMMI**

L'organisation se situe clairement au niveau 1 (Initial), marqué par des processus peu définis, dépendants des individus et sans formalisation ni standardisation. Elle doit tendre vers les niveaux supérieurs en instaurant des processus répétables, mesurables et optimisables pour garantir la qualité et la performance durable.

- **Par rapport à ITIL**

L'organisation ne dispose pas d'un cadre formel pour gérer les changements et les déploiements logiciels ce qui augmente les risques opérationnels et réduit la capacité à documenter, évaluer et améliorer les services informatiques.

L'analyse croisée avec ces modèles montre que l'organisation gagnerait à s'inspirer de bonnes pratiques issues de plusieurs référentiels, en combinant une structuration progressive (inspirée du CMMI), une gouvernance des services (ITIL) et une meilleure agilité opérationnelle (Méthodes agiles) adaptée à son contexte.

Section 3 : Proposition d'améliorations

Afin de remédier aux lacunes identifiées dans le processus de développement logiciel il est indispensable de mettre en place des solutions ciblées, adaptées aux spécificités de Naftal et aux attentes des parties prenantes. Ces propositions visent à améliorer non seulement la structuration des processus mais aussi la collaboration entre les acteurs, la qualité des livrables et la performance globale des projets. Elles s'appuient sur des pratiques reconnues issues de référentiels tels que le CMMI, les méthodes agiles et ITIL, afin d'apporter des améliorations progressives, réalistes et durables.

3.1. Les attentes des parties prenantes

Les réponses recueillies lors des entretiens ont mis en évidence plusieurs axes prioritaires d'amélioration : l'ajout d'outils de suivi de projet, la documentation des projets et la gestion plus rigoureuse de la planification. Chacune de ces propositions répond à des attentes précises exprimées par les différentes parties prenantes.

- **Mise en place d'outils de suivi de projet**

L'introduction d'outils de suivi de projet répond à une attente forte des équipes de développement et des responsables de projets. Actuellement, l'absence de support dédié rend difficile la planification, la coordination et le suivi précis des tâches. Leur adoption constitue une première étape concrète vers une gestion structurée du développement logiciel.

- **Formalisation de la documentation des projets**

Documenter systématiquement les différentes phases des projets de la définition des besoins jusqu'au déploiement permettrait de pallier un manque structurel relevé par plusieurs interlocuteurs. Les échanges oraux en réunion ne suffisent pas. Sans support écrit les décisions se perdent et les équipes peinent à capitaliser sur les expériences passées, en particulier lors de l'arrivée de nouveaux collaborateurs ou du départ de membres clés.

- **Renforcement de la planification des tâches**

La planification actuelle repose en grande partie sur des méthodes informelles. Cette situation génère des imprécisions, des chevauchements de tâches et des retards difficilement anticipés. L'une des attentes principales exprimées par les responsables est d'obtenir une vision claire et prévisible des livrables.

Ces propositions ne sont pas de simples améliorations techniques, elles constituent des leviers stratégiques permettant de répondre aux besoins spécifiques de chaque acteur impliqué, tout en renforçant la cohérence et l'efficacité globale du processus de développement logiciel.

3.2. Proposition des solutions adaptées

- **Adoption de méthodes agiles**

La mise en place de méthodes agiles, comme Scrum ou Extreme Programming (XP), permettrait d'instaurer un cadre structuré et itératif, basé sur des cycles courts (sprints), des réunions régulières (daily meetings, revues, rétrospectives) et une répartition claire des rôles (Product Owner, Scrum Master, équipe de développement). Cela faciliterait la collaboration, améliorerait la réactivité face aux besoins changeants des clients et renforcerait la qualité des livrables grâce à des feedbacks fréquents.

- **Mise en place d'outils de gestion**

L'adoption d'outils de gestion de projet et de développement collaboratif comme Jira pour le suivi des tâches et des sprints, Git pour la gestion du code source et Confluence pour la documentation partagée offrirait un support concret pour planifier, suivre et documenter les projets. Ces outils apporteraient une meilleure visibilité sur l'avancement des projets, faciliteraient la coordination entre les équipes et garantiraient une traçabilité complète des décisions et des taches réalisés.

- **Définition de standards communs et bonnes pratiques**

L'élaboration de standards communs pour la conception, le codage et les tests permettrait d'unifier les pratiques entre les différentes équipes réduisant ainsi les risques d'incohérence. Ces standards pourraient s'inspirer des référentiels existants tels que ISO/IEC 25010 ou les guidelines du Clean Code et être adaptés aux spécificités et aux contraintes locales de l'entreprise.

- **Renforcement de la formation et clarification des rôles**

Pour accompagner ces évolutions il est essentiel de prévoir des sessions de formation adaptées afin de sensibiliser les équipes aux nouvelles méthodes, outils et standards mis en

place. De plus, une clarification des rôles et responsabilités permettrait de mieux répartir les tâches pour éviter les zones d'ambiguïté et de renforcer la collaboration.

- **Intégration des indicateurs clés de performance**

L'introduction de indicateurs clés de performance (KPI) constitue une démarche structurante. En effet, la mise en place de métriques claires et partagées permettrait de suivre de manière régulière l'efficacité des actions entreprises tout au long du cycle de développement. Ces indicateurs pourraient porter sur plusieurs dimensions stratégiques tels que le respect des délais, le taux de correction des anomalies dans les délais ou encore le niveau de satisfaction des utilisateurs internes. Simples à visualiser, notamment à travers des tableaux de bord dynamiques, ils facilitent la détection rapide des écarts et l'adaptation continue des pratiques. En somme, le déploiement des KPI procure à l'entreprise un véritable instrument de gouvernance et de responsabilisation collective.

- **Intégration de la démarche OKR**

L'introduction de la méthode OKR permettrait de mieux aligner les efforts des équipes de développement avec les objectifs stratégiques de l'entreprise. Chaque équipe pourrait ainsi définir des objectifs ambitieux mais mesurables accompagnés de résultats clés vérifiables. Cette approche renforcerait la transparence, la responsabilisation et la motivation des équipes tout en facilitant l'évaluation continue des progrès réalisés.

À titre illustratif, plusieurs OKR peuvent être envisagés pour accompagner cette démarche :

- Objectif 1 : Améliorer la traçabilité et la planification des projets

Cet objectif vise à renforcer le pilotage des projets en garantissant une meilleure visibilité sur l'avancement des tâches et la répartition des responsabilités ainsi le respect des délais. Il répond à un besoin de transparence et d'efficacité opérationnelle.

Les Résultats clés :

Résultat Clé (KR)	Indicateur de performance	Délai cible
KR1 : 100 % des projets actifs intégrés dans un outil de suivi (ex. Asana ou Jira)	Nombre de projets référencés dans l'outil / Total de projets actifs	3 mois
KR2 : 90 % des tâches affectées avec une date limite et un responsable identifié	Taux de complétude des affectations dans l'outil	2 mois
KR3 : Réduction de 30 % des retards liés à une mauvaise coordination	Comparaison des retards avant/après déploiement	6 mois

Plan d'Action Détaillé :

Ressources nécessaires	Responsabilités	Indicateurs de performance
Technologies : Outil de gestion de projet (ex. Asana, Jira), intégrations avec d'autres systèmes SI si nécessaire	Chef de projet digitalisation : coordination générale	Taux d'intégration des projets dans l'outil
Humains : Chef de projet transformation digitale, administrateur Jira/Asana, référents métiers	Responsables d'équipes : saisie et suivi des tâches	Taux de tâches planifiées avec date/responsable
Financier : Licence d'outils, formation, accompagnement au changement	DSI : mise en œuvre technique de l'outil	Nombre de retards enregistrés par mois
		Satisfaction des équipes (via un sondage interne)

Étapes de Mise en Œuvre

Étape	Description	Mois
1	Audit des pratiques existantes et choix de l'outil (Jira, Asana…)	Mois 1
2	Paramétrage de l'outil (structures de projet, rôles, modèles)	Mois 2
3	Formation des équipes et phase pilote sur 2–3 projets	Mois 3

4	Déploiement global de l'outil sur tous les projets actifs	Mois 4
5	Suivi des performances, ajustements, et support utilisateurs	Mois 5-6

Implémentation et Suivi

Réunions de pilotage mensuelles pour suivre l'avancement et résoudre les blocages
Tableaux de bord dynamiques pour visualiser la progression des projets et l'affectation des tâches et les retards
Feedback des utilisateurs en continu via des sondages rapides ou ateliers participatifs

Évaluation et Amélioration Continue

Analyse comparative des performances avant/après l'adoption de l'outil
Bilan des retards réduits, réactivité améliorée, et clarté organisationnelle
Mise à jour des pratiques (ex : revoir la granularité des tâches ou l'usage des deadlines) à chaque cycle projet
Extension de la méthode à d'autres départements si résultats concluants

- Objectif 2 : Renforcer la qualité du code et des livrables

Cet objectif s'inscrit dans une démarche d'amélioration continue du processus de développement logiciel. Il vise à garantir des livrables fiables, maintenables et conformes aux bonnes pratiques techniques. Ce renforcement de la qualité est essentiel pour réduire les coûts de maintenance et améliorer la satisfaction des utilisateurs finaux.

Les Résultats clés :

Résultat Clé (KR)	Indicateur de performance	Délai cible
KR1 : Adoption d'un guide de codage par 100 % des développeurs	Taux de signature ou d'engagement au guide / Effectif total des développeurs	2 mois
KR2 : Mise en place de la revue de code obligatoire pour 100 % des fonctionnalités critiques	Pourcentage des livraisons critiques ayant une trace de revue de code	3 mois
KR3 : Réduction de 40 % des bugs détectés en production	Comparaison du nombre de bugs avant/après mise en œuvre	6 mois

Plan d'Action Détaillé :

Ressources nécessaires	Responsabilités	Indicateurs de performance
Intervenants : Formateurs internes ou externes sur les outils et l'agilité	Chef de projet ou responsable RH : planification des formations	Taux de participation aux sessions
Outils : Plateformes de formation (présentiel ou visio)	Scrum Master ou coach agile : animation et suivi des formations agiles	Fréquence des consultations de l'espace documentaire
Infrastructure : Mise en place d'un espace documentaire partagé (ex. Confluence, SharePoint)	Équipe IT : suivi des statistiques d'accès à la documentation	Feedback post-formation

Étapes de Mise en Œuvre

Étape	Description	Mois
1	Diagnostic des besoins en formation (outils + méthodes)	Mois 1
2	Planification et organisation des sessions (Git, Jira, etc.)	Mois 2
3	Formation des équipes à la méthode agile utilisée (Scrum, hybride, etc.)	Mois 3
4	Sensibilisation à l'utilisation de la documentation	Mois 3-4
5	Suivi régulier de l'usage et réévaluation des besoins	Mois 4-6

Implémentation et Suivi

Suivi des présences aux sessions de formation
Questionnaires post-formation pour évaluer la compréhension et l'utilité
Analyse des journaux d'accès à la documentation (consultation réelle vs cible)
Amélioration continue des supports de formation et de la documentation en fonction des retours

Évaluation et Amélioration Continue

Comparaison des compétences avant/après (via auto-évaluation ou test)
Évaluation de l'impact sur la productivité et l'autonomie
Mise à jour de la documentation selon les nouvelles pratiques internes ou outils adoptés
Institutionnalisation des formations

- Objectif 3 : Favoriser l'apprentissage et l'autonomie des équipes

Cet objectif vise à renforcer les compétences collectives et individuelles, améliorer l'auto-organisation des membres, et consolider une culture de partage des connaissances et de responsabilisation.

Les Résultats clés :

Résultat Clé (KR)	Indicateur de performance	Délai cible
KR1 : Organisation d'au moins 2 sessions de formation sur les outils (Git, Confluence, Jira)	Nombre de sessions organisées et taux de participation	3 mois
KR2 : 100 % des membres d'équipe formés à la méthode agile ou hybride utilisée	Taux de couverture des formations agiles	4 mois
KR3 : Mise en place d'un espace documentaire consulté au moins une fois par semaine par chaque membre	Taux de consultation hebdomadaire par utilisateur (logs)	4 mois

Plan d'Action Détaillé :

Ressources nécessaires	Responsabilités	Indicateurs de performance
Plateforme de gestion du code (GitHub, GitLab), outil d'analyse de qualité (SonarQube, ESLint…)	Architecte logiciel : Élaboration et validation du guide	Taux de couverture du guide
Humaines : Architecte logiciel, développeurs expérimentés, équipe QA	Scrum Masters / Chefs de projet : Intégration des revues de code dans le cycle de développement	Nombre de revues effectuées
Organisationnelles : Temps dédié à la rédaction du guide, revues de code intégrées aux sprints	Développeurs : Application des normes et participation active aux revues	Nombre de bugs en production (par mois ou par release)

Étapes de Mise en Œuvre

Étape	Description	Mois
1	Rédaction du guide de codage en concertation avec les développeurs seniors	Mois 1
2	Validation du guide et communication officielle auprès des équipes	Mois 2
3	Formation des équipes sur les normes de codage et les outils de revue	Mois 2
4	Intégration obligatoire de la revue de code dans le cycle de validation des fonctionnalités critiques	Mois 3
5	Suivi de la qualité du code et mesure des bugs en production	Mois 4–6

Implémentation et Suivi

Suivi via outils de versionnage : Validation des pull requests contenant la revue
Rapports mensuels sur la fréquence des revues et la couverture des bonnes pratiques
Réunions de retour d’expérience (Rétrospectives) pour ajuster le guide et le processus
Audit qualité périodique des livrables (revue aléatoire de commits ou builds)

Évaluation et Amélioration Continue

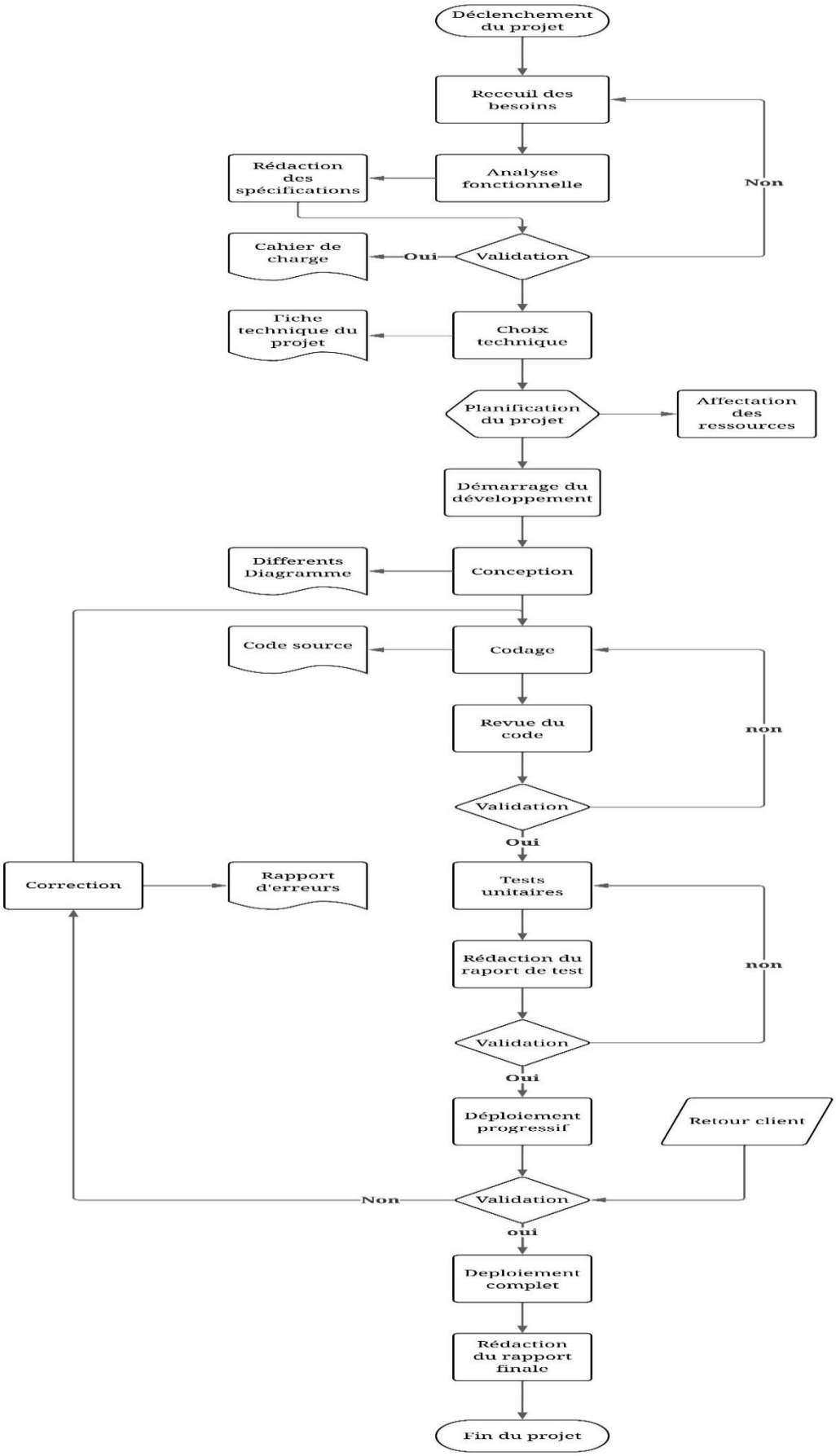
Comparaison des bugs : Nombre d’incidents de production avant/après mise en œuvre
Sondage auprès des développeurs pour mesurer l’adhésion et la satisfaction
Mise à jour du guide tous les 6 mois pour intégrer les nouvelles pratiques ou outils
Extension de la revue de code à toutes les fonctionnalités, pas seulement critiques, à moyen terme

- **Adoption progressive de l'approche DevOps**

Dans une seconde phase de structuration du processus de développement l'intégration de la culture DevOps pourrait venir compléter les pratiques agiles déjà mises en œuvre. En favorisant une collaboration renforcée entre les équipes de développement (Dev) et d'exploitation (Ops), cette approche permettrait d'automatiser des étapes clés telles que les tests, l'intégration continue (CI) et le déploiement continu (CD). Elle contribuerait ainsi à accélérer les cycles de mise en production mais aussi à réduire les erreurs humaines et à renforcer la stabilité des systèmes. Toutefois, la mise en œuvre de DevOps implique une transformation culturelle progressive, appuyée par l'adoption d'outils spécialisés tels que Jenkins, Docker ou Kubernetes.

Dans la continuité des solutions proposées, un flowchart a été élaboré grâce à Lucidchart qui est une plateforme de création de diagrammes, afin de donner une traduction concrète aux orientations d'optimisation envisagées. Ce schéma ne cherche pas à représenter absolument toutes les étapes possibles, ni à être parfait ou définitif. Construit à partir des dysfonctionnements identifiés sur le terrain, mais aussi inspiré des apports des méthodes agiles, des outils collaboratifs et des référentiels de bonnes pratiques ainsi il propose une structuration plus lisible et plus cohérente du processus de développement logiciel. Ce flowchart peut servir d'une première base qui sera évolutive et sur laquelle l'entreprise pourra s'appuyer pour initier une transformation progressive et adaptée à ses réalités.

Figure 9 : Flowchart du processus de développement logiciel visé



Source : Elaboration personnelle à partir des réponses d'entretiens

Section 4 : Discussion des résultats

Les sous questions formulées au départ de cette recherche visaient à proposer des leviers concrets d'optimisation du processus de développement logiciel chez Naftal, en tenant compte à la fois des contraintes observées sur le terrain et des bonnes pratiques identifiées dans la littérature. L'analyse croisée des données empiriques et des cadres théoriques permet aujourd'hui de revenir sur la pertinence et la portée de ces sous questions.

Concernant la première sous question qui stipule que l'introduction progressive de la méthode Scrum en combinaison avec l'approche DevOps chez Naftal permettrait de structurer le processus de développement logiciel, de renforcer la collaboration interdisciplinaire et d'améliorer la qualité des livrables et de raccourcir les délais de livraison.

Les résultats de terrain confirment l'intérêt d'introduire progressivement la méthode Scrum, en l'adaptant aux réalités locales. La structuration en sprints et l'organisation des réunions régulières ainsi que la répartition des rôles entre Product Owner et développeurs, ont été perçues comme des éléments susceptibles de répondre aux problèmes récurrents de gestion des priorités et de retard dans les livrables. De plus, la perspective d'intégrer une culture DevOps pourrait réduire les erreurs humaines et accélérer les cycles de livraison. Toutefois, les ressources humaines limitées et la faible maturité des pratiques actuelles nécessitent une mise en œuvre progressive.

La deuxième sous question relative à la mise en place d'un cadre de gouvernance agile basé sur les Objectives and Key Results (OKR) trouve également un avis favorable dans les retours d'expérience des équipes. L'absence d'indicateurs stratégiques partagés entre les niveaux hiérarchiques a été évoquée à plusieurs reprises comme un facteur de désalignement. L'approche OKR, en définissant des objectifs mesurables et compréhensibles à tous les niveaux de l'organisation permettrait de réconcilier la stratégie globale de l'entreprise avec les efforts opérationnels des équipes informatiques. Cette méthode déjà approuvée par plusieurs grandes entreprises technologiques favorise la transparence et la responsabilisation. Elle contribuerait à renforcer la motivation des collaborateurs tout en facilitant le pilotage par objectifs. Toutefois, son adoption suppose un travail de pédagogie et une discipline rigoureuse dans le suivi des résultats clés.

Quant à la troisième sous question, elle s'inscrit dans une dynamique de professionnalisation de la gestion des projets informatiques grâce à la mise en place d'indicateurs de performance.

Le terrain a révélé une absence quasi totale d'indicateurs de performance formels. Le suivi est actuellement empirique et dispersé entre les différents acteurs et rarement documenté. L'introduction de KPI adaptés tels que le taux de défauts post-livraison, le temps de cycle ou encore la satisfaction client permettrait de combler le manque de visibilité. Ces indicateurs constitueraient un socle d'évaluation facilitant la prise de décision et l'ajustement des priorités en temps réel. L'implémentation de tableaux de bord simples et visuels serait un premier pas vers une gestion plus proactive des projets.

Enfin, la quatrième sous question aborde un point souvent sous-estimé dans les projets de transformation numérique : l'adhésion des équipes. Les résistances au changement sont inévitables surtout lorsqu'il s'agit de revoir des pratiques installées de longue date. La formation, lorsqu'elle est accompagnée et contextualisée peut devenir un levier de confiance et d'autonomie. Elle favorise l'appropriation des outils et des méthodes tout en réduisant l'incertitude. De plus, l'accompagnement au changement via des coachs agiles par exemple apparaît comme un facteur clé de succès.

CONCLUSION

En conclusion de ce travail, il convient de rappeler que l'objectif principal était d'analyser comment formaliser progressivement le processus de développement logiciel dans une organisation publique algérienne en vue d'améliorer à la fois la performance et la coordination.

L'étude a mis en lumière que l'optimisation du développement logiciel ne peut se réduire à une simple adoption de méthodes ou d'outils. Elle nécessite une transformation profonde des pratiques managériales et des relations entre les parties prenantes. En contexte public, où les processus sont souvent peu formalisés et où la culture organisationnelle résiste au changement, la progression doit être progressive, appuyée par des leviers organisationnels clairs, une montée en compétence des équipes ainsi qu'un accompagnement méthodologique rigoureux.

Par ailleurs, la gouvernance par les OKR s'est révélée être un cadre pertinent pour aligner les actions opérationnelles avec la stratégie globale, tout en favorisant la transparence et la responsabilisation. Ce double mouvement de structuration et d'agilité offre un cadre capable de concilier rigueur et flexibilité, conditions indispensables pour relever les défis actuels du développement logiciel.

Concernant l'introduction progressive de Scrum et DevOps, combinée à une gouvernance par objectifs clairs et mesurables, peut engendrer une amélioration tangible de la qualité des livrables, de la communication inter-équipes et de la satisfaction des clients internes. Toutefois, la réussite de cette démarche requiert un engagement fort de la direction et un pilotage continu pour dépasser les résistances et intégrer durablement les bonnes pratiques.

Cette étude présente néanmoins certaines limites, notamment son approche qualitative centrée sur un contexte organisationnel spécifique, ce qui limite la portée des conclusions. Pour approfondir ces résultats, il serait pertinent de recourir à une méthodologie quantitative, en s'appuyant sur des indicateurs de performance précis et un échantillon plus large d'organisations publiques algériennes.

Ce travail ouvre ainsi des perspectives intéressantes pour repenser le développement logiciel dans des environnements complexes et peu structurés en conciliant innovation méthodologique et contraintes organisationnelles. Il encourage à poursuivre l'exploration des interactions entre agilité, gouvernance et maturité des processus afin d'accompagner efficacement la transformation numérique des institutions publiques.

À cet égard, il serait judicieux d'évaluer quantitativement l'impact des méthodes agiles et de la gouvernance par OKR sur la performance des projets. Par ailleurs, une analyse approfondie des facteurs humains tels que la gestion des résistances au changement et le développement des compétences pourrait révéler des clés essentielles à une adoption réussie de l'agilité dans le secteur public algérien.

De plus, l'étude de l'intégration des technologies émergentes (automatisation, intelligence artificielle…) offre un potentiel d'optimisation du pilotage des projets et d'amélioration de l'efficacité des processus. Enfin, étendre cette recherche à d'autres entités publiques ou privées en Algérie permettrait d'élaborer un cadre adaptable aux spécificités locales, renforçant ainsi la pertinence et l'applicabilité des recommandations.

Références Bibliographiques

1. Aissaoui, N. O., Layeb, S. B., Zeghal, F. M., Hamouda, C., Moujahed, H., Zaidi, A., & Jmal, Y. (2022). Amélioration de la performance d'un service d'urgences : apport du business process management et du lean management. *Revue Française de Gestion Industrielle Vol.36 Num.02*, pp. 49-56.

2. Aktouf, O. (1987). *Méthodologie des sciences sociales et approche qualitative des organisations : une introduction à la démarche classique et une critique.* Montréal: Presses de l'Université du Québec.

3. Amazon Web Services. (s.d.). *Qu'est-ce que le cycle de vie du développement logiciel (SDLC) ?* Consulté le 05 05, 2025, sur Amazon Web Services: https://aws.amazon.com/fr/what-is/sdlc/

4. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., . . . Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study. *International Conference on Software Engineering (ICSE 2019)* (pp. 291-300). Montreal: IEEE Computer Society.

5. Barrand, J. (2009). *Le Manager agile: Vers un nouveau management pour affronter la turbulence.* Paris: Dunod.

6. Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifeste pour le développement Agile de logiciels.* Récupéré sur AgileManifesto.org: https://agilemanifesto.org/iso/fr/manifesto.html

7. Bennett , N., & Lemoine, J. (2014). What VUCA Really Means for You. *Harvard Business Review*.

8. Boukhedimi, F. Z., Zerrouki, K., & Merad, M. A. (2023). Les pratiques managériales dans les organisations agiles en Algérie : Etat des lieux. *Journal of Social Protection Research Vol.04 Num.02*, pp. 38-59.

9. Cattan, M., Idrissi, N., & Knockaert, P. (1998). *Maîtriser les processus de l'entreprise : guide opérationnel.* Paris : Editions d'Organisation.

10. Center for Technology in Government. (1998). *A Survey of System Development Process Models.* Récupéré sur Center for Technology in Government, University at Albany: http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf

11. Chakir, A., Medromi, H., & Sayouti, A. (2012). La gouvernance du système d'information à base des bonnes pratiques d'ITIL V3. *JDTIC.*

12. Charbi, B., & Guesmi, A. (2021). Les méthodes de gestion de projet «agiles». *Sport system journal Vol.08 Num.01*, pp. 186-198.

13. Chrissis, M., Konrad, M., & Shrum, S. (2003). *Cmmi: Guidelines for Process Integration and Product Improvement.* Addison-Wesley.

14. Etasse , É., & Rousseau, J.-R. (2023). *Vers un développement agile.* Récupéré sur Com-Hom: http://www.com-hom.com/Fiches/0904_Vers_Un_Developpement_Agile.pdf

15. Gauthier, B. (2009). *Recherche sociale. De la problématique à la collecte des données 5e éd.* Québec: Presses de l'Université du Québec.

16. Gavard-Perret, M.-L., Gotteland, D., Haon, C., & Jolibert, A. (2012). *Méthodologie de la recherche en sciences de gestion : Réussir son mémoire ou sa thèse.* Montreuil: Pearson.

17. Ghezzi, C., Jazayeri, M., & Mandrioli, D. (1991). *Fundamentals of software engineering.* New Jersey: Prentice Hall.

18. Gravier, W. (2018). *Les 10 pratiques pour adopter une démarche DevOps efficace.* France: POESI.

19. Grosjean, J.-C. (2003). *Le Management Agile au Quotidien.* Récupéré sur Eveil Agile: https://www.eveilagile.com/management-agile-au-quotidien/

20. Highsmith, J. (2002). *Agile Software Development Ecosystems.* Boston: Addison-Wesley.

21. Horney, N. (2013). *Agility Research: History and Summary.* Agility Consulting & Training.

22. Krebs, G. (2004). *Ressources humaines : nouvelles pratiques selon l'ISO 9001.* Paris: Éditions AFNOR.

23. Melloud, S. (s.d.). *Méthodologie de recherche scientifique en sciences managériales.* Consulté le 04 25, 2025, sur ensm: https://ensm.dz/polycopie/

24. Microsoft Corporation. (s.d.). *Qu'est-ce que DevOps ?* Consulté le 04 21, 2025, sur Microsoft Azure: https://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-is-devops

25. Morley, C., Bia-Figueiredo, M., & Gillette, Y. (2011). *Processus métiers et systèmes d'information : Gouvernance, management, modélisation.* Paris: Dunod.

26. Naftal. (s.d.). *À propos de Naftal*. Consulté le 04 21, 2025, sur Naftal : https://www.naftal.dz/fr/index.php/a-propos-de-naftal

27. Naftal. (s.d.). *Moyens de Naftal*. Consulté le 04 21, 2025, sur Naftal: https://www.naftal.dz/fr/index.php/moyens

28. Naftal. (s.d.). *Produits*. Consulté le 04 21, 2025, sur Naftal: https://www.naftal.dz/fr/index.php/produits

29. Plan, F. (2024). *Les Méthodes Agiles en entreprise : quel avenir pour la fonction managériale ?* Paris: L'Harmattan.

30. Sanchez, R. (2004). Understanding Competence-Based Management: Identifying and Managing Five Modes of Competence. *Journal of Business Research Vol.57 Num.05*, pp. 518–532.

31. Schwaber , K., & Sutherland, J. (2020). *Le Guide Scrum - Le guide de référence de Scrum : Les règles du jeu.* Récupéré sur Scrum.org: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-French.pdf

32. Serehane, L., & Talbi, A. (2015). Amélioration de la performance par la mise en place de l'approche processus et la gestion des compétences. *2ème édition du Congrès International du Génie Industriel et Management des Systèmes (CIGIMS'15)*, (pp. 1-15). Fes.

33. Sutherland, J., Jakobsen, C. R., & Johnson, K. (2009). Scrum et CMMI Niveau 5 : La potion magique pour les guerriers du code Version française traduite par F. Aimetti. *International Conference on System Sciences (HICSS 2008)*, (pp. 466-466). Waikoloa.

34. Toumi Amara, D., & Kouloughli, A. (2021). L'approche processus une vision transversale au profit de l'optimisation de la formation Etude de cas : « Caisse Nationale des Retraites ». *Revue Algérienne de Développement Vol.09 Num.02*, pp. 417-430.

35. Tremblay, R. (2007). Implantation d'une méthode agile de développement logiciel en entreprise : une culture accueillant le changement. *Mémoire de maîtrise – Université Laval*. Québec, Québec, Canada: Université Laval.

ANNEXE A – GUIDE D'ENTRETIEN

Guide d'entretien

Ce guide d'entretien a été élaboré dans le but d'analyser l'état actuel du processus de développement logiciel au sein de **Naftal**. Il vise à recueillir des données qualitatives auprès des parties prenantes impliquées dans le développement logiciel (développeurs, chefs de projet, etc.) afin de mieux comprendre les pratiques en vigueur, les difficultés rencontrées ainsi que les modes de collaboration existants. Ce guide cherche également à faire émerger des pistes concrètes d'optimisation en identifiant les leviers d'amélioration possibles en matière d'organisation, de méthode et d'outils.

Informations générales

1. **Nom et fonction du répondant**
2. **Service / Département**
3. **Années d'expérience dans l'entreprise et dans le domaine du développement logiciel**

1. Organisation et Méthodologies

4. Pouvez-vous décrire le processus actuel de développement logiciel dans votre entreprise ? (De la demande initiale jusqu'à la mise en production)

5. Quelle méthodologie de développement est utilisée (Waterfall, Agile, Scrum, autre) ? Pourquoi ce choix ?

6. Existe-t-il une documentation formalisée de chaque étape du processus ? Si oui, comment est-elle faite ?

7. Comment les acteurs impliqués sont-ils organisés (équipes, rôles, responsabilités) ?

8. Comment sont gérées les demandes de changement ? (Au cours du processus de développement)

2. Outils et Technologies

9. Quels outils et langages sont utilisés pour le développement ?

10. Quels outils sont utilisés pour la gestion du cycle de vie du développement logiciel (ex. Jira, Asana, autres) ? Répondent-ils aux besoins de l'équipe ?

11. Y a-t-il des étapes du processus qui sont automatisés (ex. tests automatisés, documentation) ?

3. Qualité, Tests et Maintenance

12. Comment la qualité du code est-elle assurée ? (Tests unitaires, norme, bonne pratique, etc.)

13. Quelle place occupent les tests dans le processus ? (Automatisés, manuels, fréquence, couverture)

14. Comment sont gérées les incidents et les bugs ? Y a-t-il un processus formel de suivi et de correction ?

4. Délais et Performance

15. Utilisez-vous une méthode pour l'estimation des délais ? Si oui la quelle ? Si non pourquoi ?

16. Quels sont les principaux obstacles ralentissant le développement logiciel ?

17. Existe-t-il des indicateurs de performance pour évaluer l'efficacité du processus ? (KPIs, temps moyen de livraison, taux de défauts, etc.) un exemple ?

5. Problèmes et Opportunités d'Amélioration

18. Avez-vous identifié des axes d'amélioration ? (Automatisation, formation, changement d'outils, etc.)

19. L'introduction de l'**IA** et des **méthodes agiles** pourrait-elle améliorer l'efficacité du processus ? Pourquoi ? Quel modèle serait le mieux adapté selon vous ? (ChatGPT pour assistance au code, GitHub Copilot, IA pour tests automatisés…)

6. Perspectives et Attentes

20. Selon vous, comment pourrait évoluer le processus de développement dans les prochaines années ?

21. Si vous pouviez changer un aspect du processus actuel, lequel serait-ce et pourquoi ?