UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

LES MÉTHODES AGILES DANS LES PROJETS AÉRONAUTIQUES

MÉMOIRE PRÉSENTÉ
COMME EXIGENCE PARTIELLE DE LA

MAÎTRISE EN GESTION DE PROJET

PAR
MARCELO AMARAL DA SILVA

NOVEMBRE 2020

TABLE DE MATIÈRES

LISTE DE FIGURES

LISTE DE TABLEAUX

REMERCIEMENTS

Je tiens à remercier particulièrement le professeur Darli Rodrigues Vieira, directeur du programme de Maîtrise en Gestion de Projets, pour les encouragements, discussions et orientations dans le cadre de cette recherche.

Je remercie aussi le professeur Christophe Bredillet pour les discussions et les contributions concernant la méthodologie de recherche.

Un merci tout particulier aussi au Professeur Alencar Bravo pour ses commentaires sur le plan du contenu et de l'approche adoptée.

Finalement, je voudrais remercier ma famille et mes amis Paulo Sergio de Lima Quattrocchi et Dreyfus Siqueira Silva pour leur constante affection et soutien.

LISTE DES ABRÉVIATIONS

Abréviation	Définition originale	Signification en français
A/C	Aircraft	Avion
AC	Advisory Circular	Consultatif circulaire
AMJ	Advisory Material Joint	Joint de matériel de conseil
ANAC	Agência Nacional de Aviação Civil	Agence nationale de l'aviation civile
ASD	Adaptive Software Development	Développement de logiciels adaptatifs
AR	Acceptance Review	Revue d'acceptation
ARP	Aerospace Recommended Practice	Pratique recommandée en aérospatiale
ATA	Air Transport Association of America	Association des transports aériens d'Amérique
BSS	Business Support System	Système de soutien aux entreprises
CASE	Computer-aided software engineering	Génie logiciel assisté par ordinateur
CCA	Common Cause Analysis	Analyse de cause commune
CDR	Critical Design Review	Revue critique de la conception
CFR	Code of Federal Regulations	Code des Régulations Fédérales

CMA	Common Mode Analysis	Analyse de mode commun
COTS	Commercial off-the-shelf (product)	(Produit) commercial prêt à l'emploi
CRR	Commissioning Result Review	Revue des résultats de mise en service
CS	Conceptual Studies	Études conceptuelles
ch	Cheval Vapeur	
DBMS	Database Management System	Systèmes de gestion de bases de données
DD	Dependence Diagram	Diagramme de dépendance
DDP	Detailed Design and Certification Phase	Phase de conception détaillée et de certification
DoD	Department of Defence (United States)	Département de la défense (États-Unis)
DSDM	Dynamic Systems Development Method	Méthode de développement de systèmes dynamiques
EASA	European Aviation Safety Agency	Agence européenne de la sécurité aérienne
ECSS	European Cooperation for Space Standardization	Coopération européenne pour la normalisation spatiale
ELR	End-of-Life Review	Revue de fin de vie

EPA	Environmental Protection Agency	Agence de Protection de l'Environnement
ERP	Enterprise Resource Planning	Progiciel de Gestion Intégré
EVO	Evolutionary Project Management	Gestion de projet évolutive
EWIS	Electrical Wiring Interconnect System	Système d'interconnexion de câblage électrique
FAA	Federal Aviation Administration	Administration fédérale de l'aviation
FAR	Federal Aviation Regulation	Règlement fédéral de l'aviation
FHA	Functional Hazard Assessment	Évaluation des risques fonctionnels
FMEA	Failure Modes and Effects Analysis	Analyse des modes de défaillance et des effets
FMES	Failure Modes and Effects Summary	Résumé des modes de défaillance et des effets
FRR	Flight Readiness Review	Revue de la préparation au vol
FTA	Fault Tree Analysis	Analyse de l'arbre de défaillance
Gs	Earth's surface gravity (unit of acceleration)	Gravité de la surface de la Terre (unité d'accélération)

GUI	Graphical user interface	Interface utilisateur graphique
HP	Horse Power (unit of power)	Cheval-vapeur (unité de puissance)
IDAL	Item Development Assurance Level	Niveau d'assurance de développement d'article
IDP	Initial Definition Phase	Phase de définition initiale
ICAO	International Civil Aviation Organization	Organisation de l'aviation civile internationale
IEEE	Institute of Electrical and Electronics Engineers	Institut d'ingénieurs en électricité et électronique
EVO	Evolutionary Value Delivery	Livraison de valeur évolutive
IID	Iterative and Incremental Development	Développement itératif et incrémental
IT	Information Technology	Technologie de l'information
JAA	Joint Aviation Authorities	Autorités conjointes de l'aviation
JAD	Joint Application Development	Développement d'applications communes
JAR	Joint Aviation Requirements	Exigences conjointes de l'aviation
JDP	Joint Definition Phase	Phase de définition conjointe
Lbs	Pounds (unit of mass)	Livres (unité de masse)

LD	Lean Software Development	Développement logiciel Lean
LRR	Launch Readiness Review	Revue de la préparation au lancement
MA	Markov Analysis	Analyse de Markov
MCR	Mission Close-out Review	Revue de clôture de la mission
MDR	Mission Definition Review	Revue de la définition de la mission
MG	Maturity Gate	Porte de maturité
MIL	Military Specifications / Requirements	Spécifications/exigences militaires
MoC	Means of Compliance	Moyens de conformité
MoSCoW	Must Have, Should Have, Could Have and Won't Have this time	Doit être fait, devrait être fait, pourrait être fait et ne sera pas fait cette fois
MTOW	Maximum Takeoff Weight	Poids maximaux au décollage
OEM	Original Equipment Manufacturer	Fabricant d'équipement d'origine
ORR	Operational Readiness Review	Revue de l'état de préparation opérationnelle
PD	Preliminary Design	Conception préliminaire
PDR	Preliminary Design Review	Revue de conception préliminaire

PHAC	Plan for Hardware Aspects of Certification	Plan des aspects de hardware de la certification
PMBOK	Program Management Body of Knowledge	Document de référence en gestion de programme
PMI	Program Management Institute	Institut de gestion de programme
PRA	Particular Risks Analysis	Analyse des risques particuliers
PRR	Preliminary Requirements Review	Revue préliminaire des exigences
PS	Preliminary Studies	Études préliminaires
PSAC	Plan for Software Aspects of Certification	Plan pour les aspects logiciels de la certification
PSSA	Preliminary System Safety Assessment	Évaluation préliminaire de la sécurité du système
QR	Qualification Review	Revue de qualification
RAD	Rapid Application Development	Développement rapide d'applications
RAF	Royal Air Force	Force Aérienne Royal
RTCA	Radio Technical Commission for Aeronautics	Commission technique radio pour l'aéronautique
RUP	Rational Unified Process	Processus unifié de Rational

S1, S2, S3	Serial number 1, Serial number 2, Serial number 3	Numéro de série 1, Numéro de série 2, Numéro de série 3
SAE	Society of Automotive Engineers	Société des ingénieurs de l'automobile
SRR	System Requirements Review	Revue des exigences du système
SSA	System Safety Assessment	Évaluation de la sécurité du système
SSADM	Structured Systems Analysis and Design Method	Méthode d'analyse et de conception de systèmes structurés
STD	Standard	Norme
TCCA	Transport Canada's Civil Aviation	Transports Canada Aviation civile
UK	United Kingdom	Royaume-Uni
US	United States of America	États-Unis d'Amérique
VFR	Visual flight rules	Règles de vol à vue
XP	Extreme Programming	Programmation extrême
ZSA	Zonal Safety Analysis	Analyse de sécurité zonale

1. INTRODUCTION

1.1. Contexte

Les Méthodes Agiles sont utilisées dans les sociétés de logiciels depuis 1986, lorsque le mot Scrum (Schwaber et al., 2016) a été utilisé pour la première fois dans un magazine « Harvard Business Review » (Takeuchi et al., 1986). Depuis lors, un grand nombre de Méthodes Agiles ont été produites jusqu'en 2001, lorsqu'un groupe de développeurs expérimentés a rassemblé les principes qui sous-tendent ces méthodes, donnant naissance au "Manifeste Agile". Le travail comprenait quatre valeurs et douze principes (Beck et al., 2001). Il a été prouvé que les Méthodes Agiles aident l'entreprise à économiser du temps et de l'argent (Baseer et al., 2015).

Les méthodes de développement Agiles prennent en charge une large gamme de cycle de vie de développement logiciel (Abrahamsson et al., 2002). Cela comprend la création de concept, la spécification des exigences, la conception, le code, le test unitaire, le test d'intégration, le test du système, le test d'acceptation et le système utilisé.

Le temps pour le développement d'un avion est considérablement long. Il faut généralement 5 à 10 ans entre le lancement du programme et le premier avion livré. À titre d'exemple, le Boeing 777 a commencé les travaux de conception préliminaire en janvier 1990. Les premiers contrats d'achat pour le premier 777 ont été signés neuf mois plus tard en octobre 1990. Le premier

prototype a volé près de 4 ans plus tard, en juin 1994, en commençant la phase de test en vol. Le premier acheteur du 777, la compagnie United Airlines, a reçu son premier avion en mai 1995.

Au fil du temps, les avions deviennent de plus en plus complexes et leur temps de développement augmente. Pour le Boeing 737 par exemple, depuis la conception préliminaire en mai 1964 jusqu'à la réception du premier avion en décembre 1967, le B737 n'a pris que 3 ans et 7 mois, contre 5 ans et 4 mois pour le B777.

Lors du développement d'un nouvel avion, c'est-à-dire avant même de produire les premiers prototypes, les entreprises commencent à acheter des avions, c'est-à-dire qu'elles signent des contrats de vente avant de voir l'avion final. À partir d'un contrat signé, s'il y a retrait de l'un des deux côtés, des pénalités et des amendes sont appliquées. Cela vous donne une idée de combien il est important qu'un nouveau programme d'avion devienne un succès dans le délai imparti.

1.2. Problème général

Il y a peu de littérature sur comment mettre en œuvre les Méthodes Agiles dans les projets d'avions.

Les Méthodes Agiles ont été utilisées de préférence dans le développement de logiciels, au lieu des méthodes traditionnelles (cascade), car elles produisent des résultats à livrer aux parties intéressées dans un délai plus court. Ces résultats peuvent ensuite être examinés, puis les prochains livrables peuvent

avoir un meilleur plan d'action. Le coût des changements peut alors être plus faible par rapport à ceux des méthodes traditionnelles.

Les méthodes de développement traditionnelles utilisées dans le développement des avions sont des méthodes très strictes où les problèmes découverts dans les phases ultérieures ne sont pas autorisés à interférer dans les phases qui sont déjà terminées. Cela provoque des problèmes découverts dans les phases avancées du projet pour faire du projet un échec. Plusieurs projets ont échoué dans l'histoire, comme le North American Aviation XB-70, qui a coûté au total plus de 10 milliards de dollars américains et seulement 2 prototypes ont été construits.

La méthode de développement la plus utilisée dans l'industrie aéronautique est la méthode en cascade, ou « waterfall » en anglais. Dans ce type de méthodologie, il devrait y avoir une vérification à la fin de chaque phase et une approbation formelle pour passer à la phase suivante. Cette approche planifiée est définie de manière rigide dans la phase d'analyse des exigences et empêche les modifications de manière très impérative. Un problème rencontré en raison d'un travail médiocre dans une phase donnée rencontré à une phase ultérieure peut augmenter les ressources utilisées à un niveau qui peut rendre impossible la poursuite du projet.

1.3. Problème spécifique

Un autre type de modèle de développement a été utilisé avec beaucoup de succès dans l'industrie du logiciel, qui s'appelle les Méthodes Agiles. L'une des plus grandes difficultés à appliquer des Méthodes Agiles dans une industrie où l'on a la culture d'employer des méthodes plus traditionnelles telles que la cascade est que les Méthodes Agiles, par leur nature même, reçoivent bien des changements à n'importe quelle phase du développement. Ce type d'approche est très bien utilisé dans l'industrie du logiciel car il semble plus facile de modifier certaines parties du logiciel que de changer une aile d'avion. Mais si vous comparez le coût de développement d'un avion au coût du développement de logiciels, pour être honnête, nous devrions considérer un logiciel aussi complexe et coûteux qu'un avion. Changer une grande fonctionnalité d'un logiciel complexe et coûteux peut être aussi coûteux et complexe que de changer la forme de l'aile d'un avion pour une raison impérative.

Une autre difficulté d'utiliser les Méthodes Agiles dans les industries traditionnelles est que, bien que vous puissiez, en théorie, fournir un logiciel manquant quelques fonctionnalités mais pouvant être utile au client, les gens se demandent comment nous pourrions livrer un avion sans certaines fonctions. Pour être honnête dans cette comparaison, tout comme un logiciel ne peut pas être livré au client sans les fonctionnalités principales, comme un éditeur de texte qui ne peut pas afficher à l'écran comment le texte final s'imprimerait, un avion ne peut pas non plus être livré sans le principales caractéristiques qui

garantissent la sécurité d'un vol. De plus, lorsque nous voyons les différentes versions produites à partir du même avion, nous voyons que l'industrie aéronautique n'est pas si loin de l'industrie des logiciels complexes où les Méthodes Agiles sont appliquées avec succès - considérant au moins les versions basées sur les anciennes versions.

Les Méthodes Agiles sont déjà présentes dans la fabrication aérospatiale (Presley et al., 1998), avec des techniques Agiles comme l'ingénierie simultanée, l'intégration en entreprise, les équipes de travail autogérées et les principes et pratiques avancés du « lean manufacturing ».

L'application de Méthodes Agiles pour l'industrie aéronautique a été un défi car ces méthodes accueillent les changements des exigences logicielles, même tard dans le développement (Beck et al., 2001), mais le développement des aéronefs ne permettre généralement aucun changement tardif des exigences.

Les méthodes Scrum peuvent encore être utilisées dans l'industrie du logiciel aéronautique, pour conduire à des résultats de meilleure qualité, mais la manière de le faire reste à définir.

L'objectif principal de cette étude est de savoir dans quelle mesure nous pouvons utiliser les Méthodes Agiles dans l'industrie aéronautique. En d'autres termes, nous essayons d'utiliser des Méthodes Agiles pour augmenter la performance de développement des avions et minimiser le coût final.

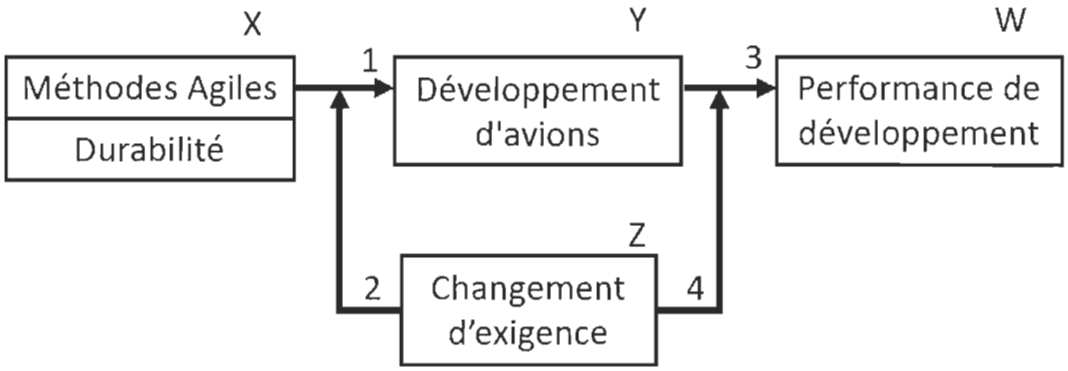

Figure 1 – Cadre conceptuel

1.4. Localisation

Le but de ce travail est de proposer une solution pour développer des avions en définissant combien nous pouvons utiliser des Méthodes Agiles dans le contexte du cadre Scrum. Nous proposons un moyen de minimiser la zone 1 et de maximiser la zone 2 (*Figure 2*).

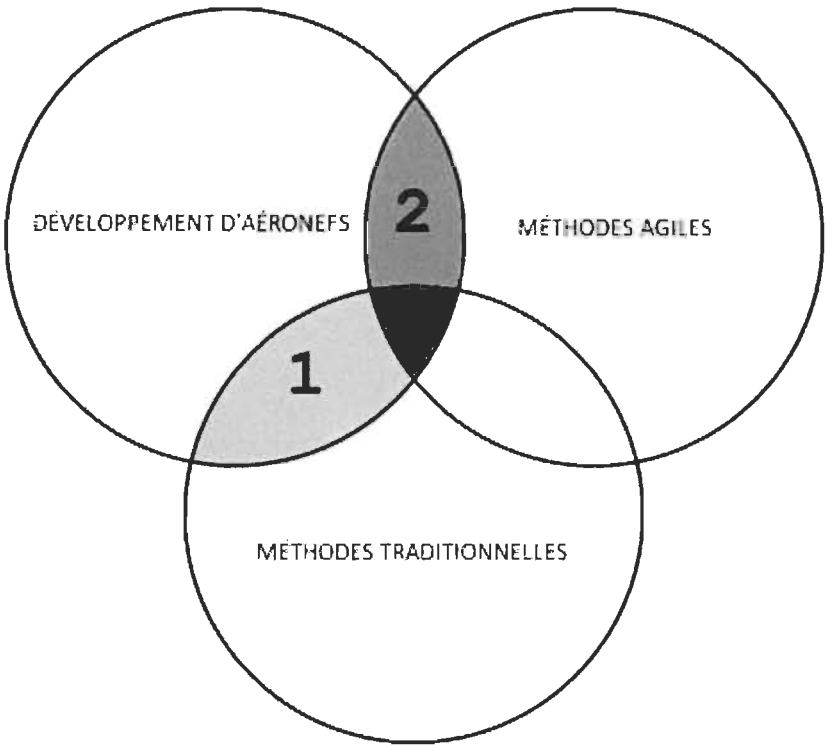

Figure 2 – Localisation de la recherche

La façon d'appliquer directement les Méthodes Agiles au développement de l'avion part de que nous avons réussi à appliquer comme méthodes de développement logiciel. Ce n'est pas facile à faire en raison de la nature différente de ces produits. Le produit logiciel peut être livré inachevé au client avec un ajout ultérieur de certaines fonctionnalités. Cela peut difficilement être fait pour le produit d'avion, car le client s'attend à ce que le produit soit dans son état de production final pour être livré.

L'approche proposée consiste à présenter chaque pratique Agile et à vérifier si elle peut être appliquée au développement de l'avion sans compromettre ou risquer les jalons du projet. Une autre approche consiste à vérifier si les méthodes

utilisées dans le développement des avions peuvent être modifiées ou adaptées aux Méthodes Agiles.

1.5. Objectifs et questions de recherche

Une série d'attributs de durabilité des logiciels a été définie pour évaluer les performances de durabilité d'un système logiciel (Albertao, 2004).

Ces attributs tels que la modifiabilité, la réutilisabilité, la portabilité, performance, la fiabilité, l'utilisabilité, entre autres, peuvent être utilisés avec les principes du Manifeste Agile pour construire une liste des aspects à surveiller pendant le sprint de développement logiciel pour construire un logiciel plus durable.

Cet article explique en termes pratiques comment le Manifeste Agile peut être utilisé pour développer des avions dans le contexte des Méthodes Agiles.

Les Méthodes Agiles sont le moyen le plus utilisé pour développer des logiciels ces derniers temps. Cette approche du développement logiciel est itérative, adaptative, bien planifiée et organisée, donne le contrôle de l'amélioration, et donne une livraison précoce et fréquente. Il a des équipes autoorganisées, ce qui donne à l'équipe la responsabilité de bien utiliser ses ressources. Tout le monde dans l'équipe connaît ses responsabilités et ils travaillent ensemble sans laisser aucune ressource inactive, donc les performances sont optimisées.

Le développement durable est une bonne pratique à utiliser telle qu'elle est choisie car nous devons développer en utilisant le moins d'effort possible.

Nous voulons être en mesure de réduire les coûts en utilisant moins d'énergie possible pour développer des avions, et faire que le résultat soit une meilleure performance de développement.

Comme il y a peu de littérature sur la façon dont nous pouvons utiliser les Méthodes Agiles pour développer des avions, ce travail présente une façon de le faire.

Pour définir une solution au problème proposé, nous avons établi le *Tableau 1* pour organiser les objectifs que nous avons et pour poser des questions sur la façon dont les objectifs peuvent être atteints.

Tableau 1 - Objectifs et questions de recherche

O1 (Xa): Définir les Méthodes Agiles. (§2.3)	RQ1 (Xa): Quelles sont les Méthodes Agiles?
O2(Xb): Définir la durabilité. (§2.3.12)	RQ2(Xb): Qu'est-ce que la durabilité?
O3 (Y): Définir le développement de l'avion. (§2.4)	RQ3 (Y): Qu'est-ce que le développement de l'avion?
O4 (Z): Définir le changement des exigences. (§2.5)	RQ4 (Z): Qu'est-ce que le changement des exigences?
O5 (W): Étudier la performance du développement. (§2.6)	RQ5 (W): Qu'est-ce que la performance du développement?
O6 (1a): Étudier les Méthodes Agiles dans le développement de l'avion. (§2.5; §4)	RQ6 (1a): Pouvons-nous appliquer les Méthodes Agiles au développement de l'avion?

O7 (1b): Étudier la durabilité dans le développement de l'avion. (§4)	RQ7 (1b): Pouvons-nous appliquer la durabilité dans le développement de l'avion?
O8 (2a): Étudier le changement des exigences dans les Méthodes Agiles. (§2.5)	RQ8 (2a): Comment le changement des exigences est traité dans les Méthodes Agiles?
O9 (2b): Étudier le changement des exigences dans la durabilité. (§2.5)	RQ9 (2b): Quel est le lien entre le changement des exigences et la durabilité?
O10 (3): Étudier le changement des exigences dans le développement de l'avion. (§2.5)	RQ10 (3): Comment le changement des exigences affecte le développement de l'avion?
O11 (4): Étudier le changement des exigences dans la performance du développement. (§2.5)	RQ11 (4): Comment le changement des exigences affecte la performance du développement?
O12 (5): Étudier la performance du développement dans le développement de l'avion. (§2.6)	RQ12 (5): Comment la performance du développement affecte le développement de l'avion.

1.6. Périmètre

La discussion se concentrera sur la comparaison des processus de développement des avions avec les Méthodes Agiles, pour vérifier ce qui est déjà utilisé, et pour définir les défis et ce qui peut être fait de plus pour appliquer les Méthodes Agiles dans le développement des avions.

Dans un premier temps, nous analyserons certaines des méthodes de développement traditionnelles utilisées dans l'industrie comme le modèle

classique de développement en cascade, le phasage du projet selon l'ECSS et le cycle de vie du développement selon la norme aéronautique ARP4754A et d'autres.

Ensuite, nous étudierons certaines des Méthodes Agiles les plus utilisées comme le développement itératif et incrémental, la gestion de projet évolutive (EVO), le Adaptive Software Development (ASD), le développement rapide d'applications (RAD), le Rational Unified Process (RUP), le Dynamic Systems Development Method (DSDM), Scrum, Extreme Programming (XP), le Crystal et le développement basé sur les fonctionnalités (FDD). Ensuite, nous examinons le Manifeste Agile, la définition de la durabilité, et résumons les pratiques des Méthodes Agiles.

Après cela, nous étudions les méthodes, le flux de documentation et les revues (reviews) que nous avons dans le développement actuel des avions et nous voyons les exemples des processus de développement des avions Airbus et Embraer.

Nous étudions également les impacts que le changement d'exigence et la performance de développement ont sur les processus de développement pour enfin discuter et proposer des techniques pour appliquer des Méthodes Agiles dans le développement des avions.

Le processus de développement des aéronefs est réglementé par les agences gouvernementales de navigabilité et les méthodologies de

développement ne sont pas clairement définies, laissant les constructeurs d'aéronefs décider de la méthodologie à utiliser.

Les Méthodes Agiles se sont révélées être un processus de développement plus durable, contrairement à la méthode traditionnelle en cascade couramment utilisée dans l'industrie aéronautique.

Ce travail a également pour objectif de présenter les moyens d'augmenter l'utilisation des Méthodes Agiles dans le processus de développement des avions.

2. REVUE DE LITTÉRATURE

2.1. Approches de développement logiciel

Le processus de développement logiciel est complexe et, même pour un petit produit, impliquant une seule personne, certaines étapes de base doivent être respectées si l'on veut avoir un bon produit.

Voici quelques étapes à suivre, même pour les logiciels les plus simples à produire:

Analyse fonctionnelle - le développeur définit quel produit il souhaite développer, en réponse à un besoin donné.

Spécification - dans cette étape, le développeur identifie toutes les fonctionnalités qui doivent être incluses dans le logiciel, également appelées analyse des exigences. La langue à utiliser dans le code est choisie comme la meilleure pour la spécification proposée.

Conception - la conception est la définition de haut niveau du produit, où les caractéristiques d'une même famille sont organisées en blocs ou en groupes. Toutes les relations entre ces blocs sont également définies dans cette étape.

Implémentation - c'est le moment d'écrire le code.

Test - le test est l'une des étapes les plus importantes pour écrire un code de qualité. Un logiciel non testé peut ne pas fonctionner comme souhaité dans un environnement non contrôlé comme celui qu'il a été utilisé pour la production. La façon ou le moment où le code est testé dépend du processus de développement logiciel choisi.

Documentation - La documentation est importante car le code est généralement corrigé après le test, et il est également généralement réutilisé.

Déploiement - cette étape correspond à l'installation et à l'exécution du code dans l'environnement client.

Maintenance - cela se produit après le déploiement lorsque le logiciel rencontre des problèmes d'utilisation. Cela se produit également lorsque le client a de nouveaux besoins. Des problèmes et de nouvelles exigences se produisent souvent, et ils sont entendus et accordés dans le contrat entre le développeur et le client.

Jusqu'à présent, nous n'avons pas parlé du processus de développement du logiciel. Toutes ces étapes doivent se produire à un moment donné du développement, quelle que soit l'approche de développement utilisé.

Les processus de développement logiciel peuvent être divisés en deux groupes différents: le modèle classique et le modèle Agile.

2.2. Modèle classique

2.2.1. Le modèle classique de développement en cascade

Nous ne pouvons pas commencer à parler des Méthodes Agiles avant de parler des méthodes qui ont été utilisées avant. Dans le modèle de développement en cascade, le processus de développement se déroule dans une direction et il est clairement défini en phases telles que l'analyse des exigences, la conception, le codage (mise en œuvre), les tests (validation) et la mise en service (*Figure 3*).

La pratique habituelle de ce modèle prétend qu'il doit avoir une vérification, ou une revue, à la fin de chaque phase, et une approbation formelle pour passer à la phase suivante (review). Une fois qu'une nouvelle phase est atteinte, en principe, on ne revient pas aux phases précédentes.

Cette approche pilotée par le plan est définie de manière rigide lors de la phase d'analyse des exigences et désapprouve de manière forcée les modifications. Un problème dû à un travail insuffisant dans une phase seulement trouvé dans une phase ultérieure peut augmenter l'utilisation des ressources à un niveau qui peut rendre le projet impossible à compléter.

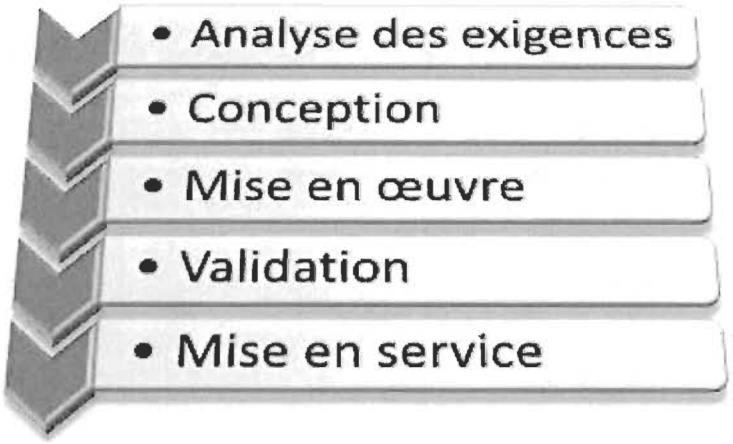

Figure 3 – Phases du modèle de développement en cascade

L'une des utilisations classiques du modèle cascade (ou Waterfall) est le SSADM (Structured Systems Analysis and Design Method - Méthode d'Analyse et de Conception de Système Structurés) produit pour le Central Computer and Telecommunications Agency, du Royaume-Uni.

Cette méthode comprend 6 étapes (Great Britain, 2000):

- Étude de l'environnement actuel,
- Options de système d'affaires,
- Définition des exigences,
- Options techniques du système,
- Conception logique,
- Conception physique.

2.2.2. Phases du projet selon ECSS

L'ECSS est la Coopération Européenne pour la Normalisation Spatiale (European Cooperation for Space Standardization), une organisation qui vise à définir des normes pour les projets spatiaux européens. Ils produisent de nombreux documents standard qui incluent la gestion de projet, l'assurance produit et les activités d'ingénierie (Goh, 2010).

Le cycle de vie du développement ECSS est très similaire à celui utilisé pour le développement de l'avion.

Le cycle de vie des projets spatiaux (ECSS-M-ST-10C, 2009), selon la Coopération Européenne pour la Normalisation Spatiale, comprend 7 phases (*Figure 4*):

- Phase 0 - Analyse de la mission / identification des besoins
- Phase A - Faisabilité
- Phase B - Définition préliminaire
- Phase C - Définition détaillée
- Phase D - Qualification et production
- Phase E - Utilisation
- Phase F – Cession

L'ECSS (ECSS-M-ST-10C, 2009), répartit les phases du projet en groupes:

• Phases 0, A et B, où les fonctions du système et les exigences techniques sont élaborées, et les objectifs de la mission sont analysés pour identifier les concepts des systèmes en fonction des contraintes techniques et programmatiques identifiées par l'initiateur du projet et le client de haut niveau. Les activités, les ressources à utiliser et les risques associés sont également identifiés (ECSS-M-ST-10C, 2009).

• Les phases C et D se concentrent sur le développement et la qualification.

• La phase E est liée au lancement, à la mise en service et au service.

• La phase F est liée à la fin de vie du produit.

Revues de projet

Les revues suivantes sont incluses dans le cycle de vie du projet selon l'ECSS (ECSS-M-ST-10C, 2009):

MDR - Revue de Définition de la Mission (Mission Definition Review)

PRR - Revue Préliminaire des Exigences (Preliminary Requirements Review)

SRR - Revue des Exigences du Système (System Requirements Review)

PDR - Revue Préliminaire de Conception (Preliminary Design Review)

CDR - Revue Critique de Conception (Critical Design Review)

QR - Revue de Qualification (Qualification Review)

AR - Revue d'Acceptation (Acceptance Review)

ORR - Revue de Préparation Opérationnelle (Operational Readiness Review)

FRR - Revue de Préparation au Vol (Flight Readiness Review)

LRR - Revue de Préparation au Lancement (Launch Readiness Review)

CRR - Revue des Résultats de Mise en Service (Commissioning Result Review)

ELR - Revue de Fin de Vie (End-of-Life Review)

MCR - Revue de Clôture de la Mission (Mission Close-out Review)

Le cycle de vie du produit est également défini par la présence d'activités liées:

- Mission/Fonction
- Exigences
- Définition
- Vérification
- Production
- Utilisation
- Cession

Les activités de la Mission/Fonction sont développées pendant la phase 0 et la phase A; La phase 0 se termine par la Revue de Définition de la Mission (MDR) et la phase A se termine par la Revue Préliminaire des Exigences (PRR).

Les activités d'Exigences sont développées au cours des phases 0, A et B; La phase B se termine par la Revue Préliminaire de Conception (PDR); la Revue des Exigences du Système (SRR) a lieu pendant la phase B.

Les activités de Définition sont développées au cours des phases B et C; La phase C se termine par la Revue Critique de Conception (CDR).

Les activités de Vérification sont développées au cours des phases C, D et E; la Revue de Qualification (QR) a lieu pendant la phase D.

Les activités de Production sont développées au cours des phases C et D; La phase D se termine par la Revue d'Acceptation (AR) et la Revue de Préparation Opérationnelle (ORR) en même temps.

Les activités d'Utilisation sont développées au cours de la phase E; les activités d'utilisation commencent par la Revue des Résultats de Mise en Service (CRR); La phase E se termine par la Revue de Fin de Vie (ELR); la Revue de Préparation au Vol (FRR) et la Revue de Préparation au Lancement (LRR) ont lieu pendant la phase E et avant le début des activités d'Utilisation.

Les activités de Cession ont lieu pendant la phase F; la phase F se termine par la Revue de Clôture de la Mission (MCR).

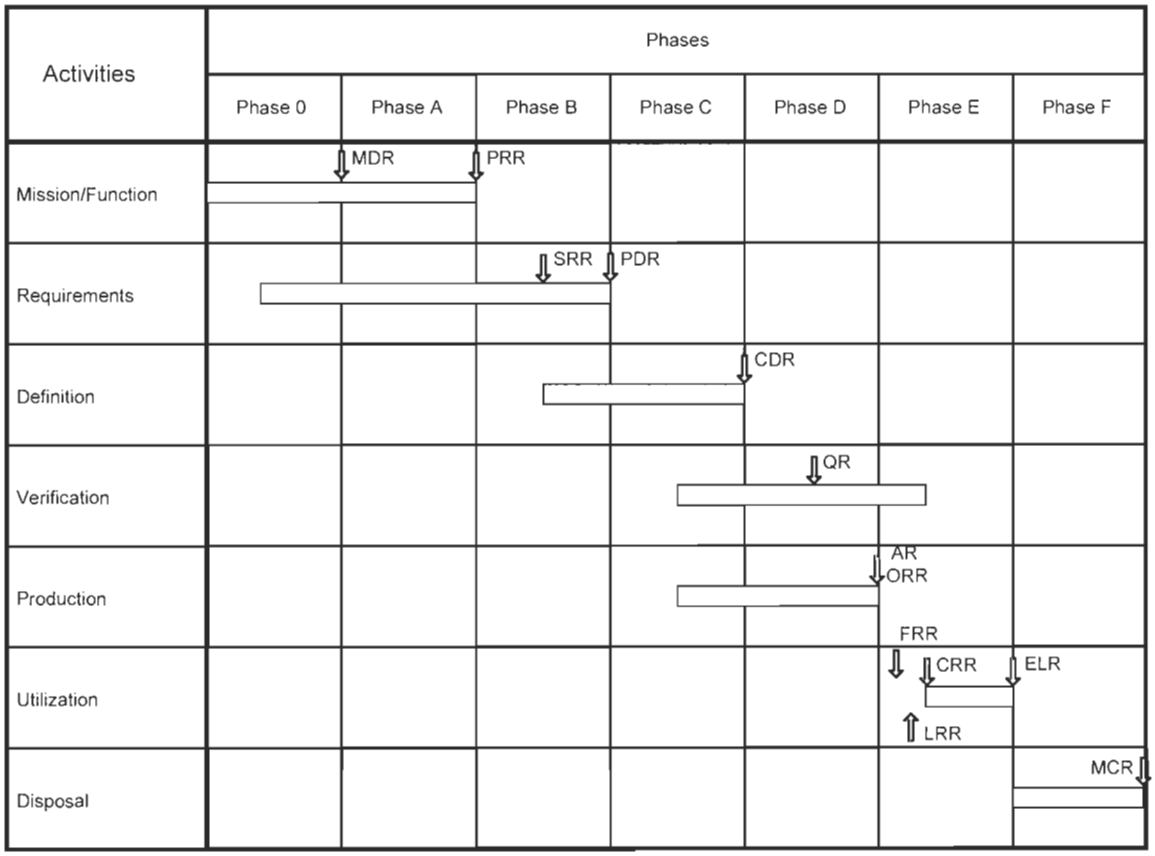

Figure 4 – Le cycle de vie du projet selon ECSS

(Crédits d'image: European Cooperation for Space Standardization, ECSS-M-ST-10C)

Le V-Model est connu comme une représentation graphique d'un cycle de vie de développement de projet. L'ECSS a défini sa propre représentation du V-Model comme nous le voyons à la *Figure 5*, et elle est appelée Revue de Cycle de Vie (Review Life Cycle) (ECSS-M-ST-10C, 2009).

Nous voyons dans le ECSS V-Model le flux des évaluations provenant de l'initiateur du projet (Project Initiator) jusqu'au fournisseur de niveau le plus bas (Lowest Level Supplier) en passant par le nombre de clients de niveau (Level

Customers), du niveau supérieur au nième, et retour du fournisseur de niveau le plus bas (Lowest Level Supplier) jusqu'à l'opérateur du système (System Operator) en passant par le nombre de fournisseurs de niveau (Level Suppliers), du niveau le plus bas au premier niveau, et en passant par un certain nombre de révisions. Il s'agit du modèle de cycle de vie V-Model du point de vue des clients, des fournisseurs et des revues.

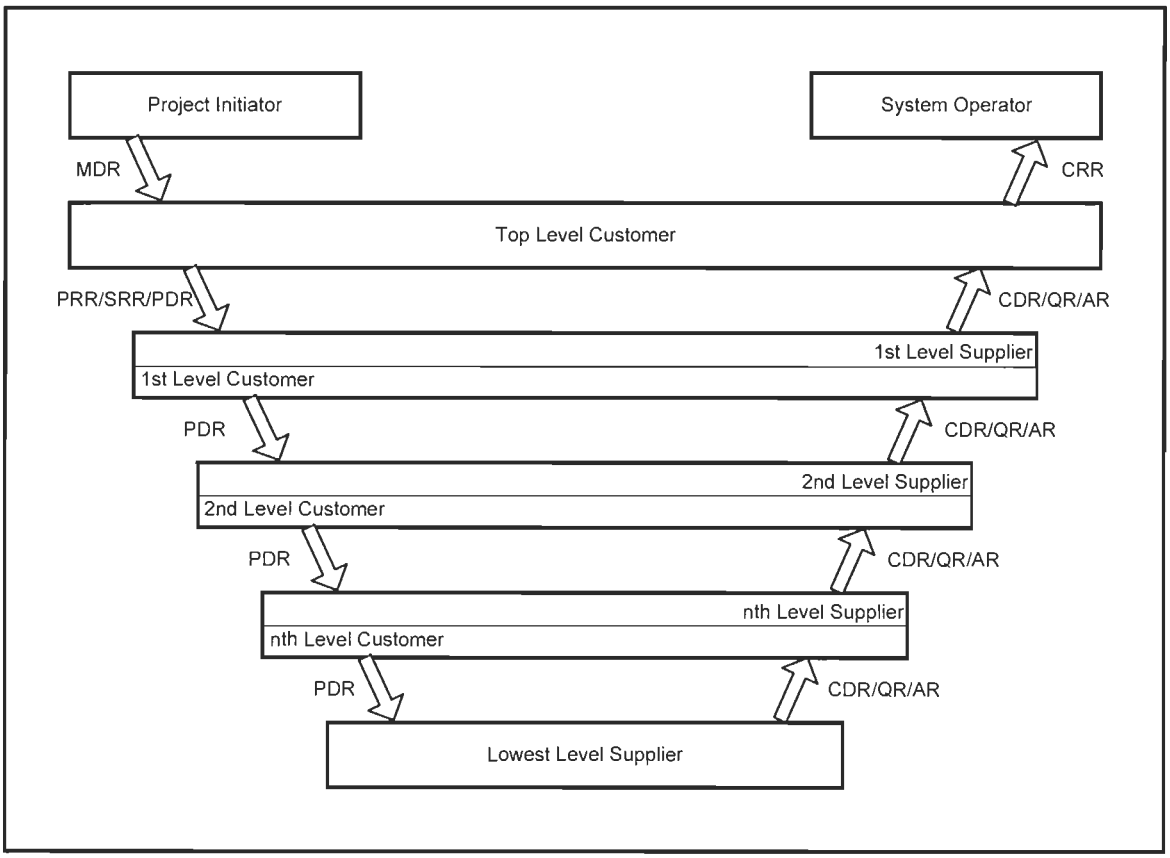

Figure 5 – Le V-Model selon ECSS

(Crédits d'image: European Cooperation for Space Standardization, ECSS-M-ST-10C)

2.3. Méthodes Agiles

Les Méthodes Agiles ont été incorporées dans l'industrie du logiciel à plusieurs niveaux, mais elles étaient pour la première fois bien définies dans le Manifeste Agile.

Depuis l'utilisation de la méthode Scrum en 1986 (Takeuchi et al., 1986), de nombreuses méthodes Agile ont été produites jusqu'en 2001 lorsqu'un groupe de développeurs chevronnés a rassemblé les principes sous-jacents à ces méthodes, donnant naissance au Manifeste Agile.

Certaines des méthodes Agile comme Scrum et RAD (Rapid Application Development) dans les années 80, et XP (Extreme Programming), Crystal, DSDM (Dynamic Systems Development Method), ASD (Adaptive Software Development) ont été produites dans les années 90 avant la publication du Manifeste Agile.

Le Manifeste Agile comprenait douze principes, dont l'un affirmait: "Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant." (Beck et al., 2001)..

Il a été prouvé que les Méthodes Agiles aident l'entreprise à économiser du temps et de l'argent (Baseer et al., 2015).

2.3.1. Développement itératif et incrémentiel

La première utilisation documentée du développement itératif et incrémental (Iterative and Incremental Development - IID) concerne le jet supersonique X-15 (1959), et son utilisation est considérée comme l'une des principales contributions au succès du projet (Larman et al., 2003).

La définition PMBOK du cycle de vie itératif est (Project Management Institute, 2017a):

"Dans le cas d'un cycle de vie itératif, le périmètre du projet est généralement déterminé au début de son cycle de vie. Les estimations des délais et des coûts sont changées régulièrement à mesure que l'équipe projet comprend mieux le produit. Les itérations développent le produit à travers une série de cycles répétitifs, tandis que les incréments ajoutent progressivement des fonctionnalités au produit."

La définition PMBOK du cycle de vie incrémentiel est (Project Management Institute, 2017a):

"Dans le cas d'un cycle de vie incrémentiel, les livrables proviennent d'une série d'itérations qui ajoutent progressivement des fonctionnalités dans une période de temps prédéterminée. Les livrables incluent les fonctionnalités nécessaires et suffisantes pour être considérés comme exhaustifs uniquement après l'itération finale."

Tandis qu'itératif signifie raffinement successif et répétitif, incrémentiel signifie un processus lorsque quelque chose est construit en parties ou en morceaux qui sont ajoutés à l'ensemble pendant le processus. Bien qu'il soit difficile de les séparer dans un projet pratique, nous pouvons penser à un projet purement itératif lorsque chaque équipe travaille sur une partie (fonctionnalité) différente et après chaque cycle, ils offrent une amélioration des fonctionnalités pour la version logicielle donnée.

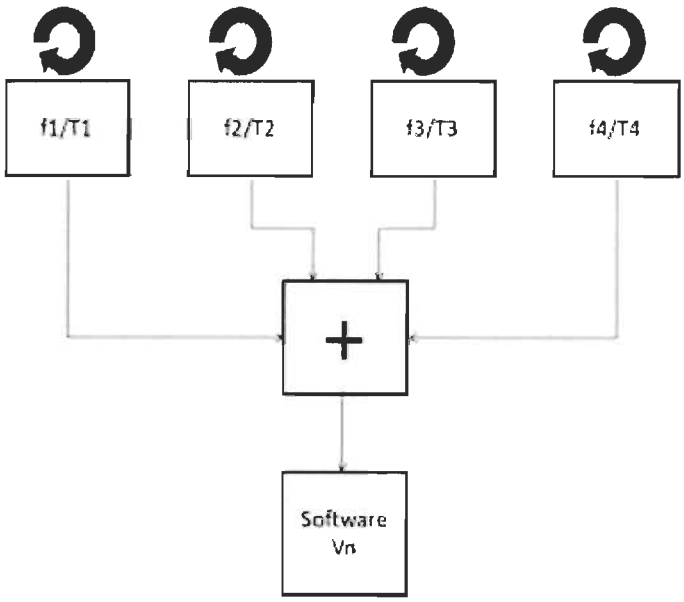

Figure 6 – Le développement purement itératif

L'équipe 1 (T1) travaille pour livrer la fonctionnalité 1 (f1), tandis que l'équipe 2 travaille pour livrer la fonctionnalité 2, et ainsi de suite. L'idée de collaboration est introduite au fur et à mesure que les équipes travaillent pour produire un

logiciel final. La concurrence est également introduite car les équipes travaillent en même temps (développement simultané ou parallèle). Si la même équipe est responsable du développement et des tests, nous avons des équipes interfonctionnelles. Les tests automatisés sont essentiels au développement itératif (Kroll et al., 2003). L'idée d'itération est fortement basée sur le retour d'information (feedback), qui devrait indiquer à l'équipe ce qu'il faut améliorer ou corriger. Les feedbacks peuvent provenir de l'intérieur de l'équipe ou des utilisateurs. Ici, l'ensemble du travail est divisé entre de plus petites équipes. Cette méthode est connu pour améliorer les performances du développement, car les équipes plus petites et les petits morceaux de travail sont plus faciles à gérer.

La première version du logiciel est le produit de la première itération, et elle a toutes les fonctionnalités intégrées. La première version du logiciel est alors:

V1 = f1 + f2 + f3 + f4

La deuxième version du logiciel a le même nombre de fonctionnalités, bien que certaines fonctionnalités soient améliorées. La deuxième version du logiciel est alors, considérant que toutes les fonctionnalités ont été améliorées:

V2 = f1' + f2' + f3' + f4'

Les versions logicielles suivantes ont au moins une fonctionnalité améliorée. Par exemple, la troisième version logicielle n'a que la fonctionnalité numéro 1 améliorée, alors ce serait:

V3 = f1'' + f2' + f3' + f4'

Une régression peut se produire et une ou plusieurs fonctionnalités peuvent revenir à un état antérieur. Par exemple, après la version logicielle V9 = f1''' + f2'' + f3' + f4'''', la fonction f4 devait retourner à l' état précédent, donc la version logicielle V10 serait:

V10 = f1''' + f2'' + f3' + f4'''

Les itérations peuvent utiliser le timeboxing ou non. Il est commun d'avoir du timeboxing, mais le projet peut également être basé sur le feedback. Larman, (Larman et al., 2003), donne de nombreux exemples de projets IID basés seulement sur le timebox, seulement sur le feedback et aussi basé sur les deux.

On peut penser à un projet purement incrémentiel lorsque chaque version logicielle reçoit une nouvelle fonctionnalité (feature). La même équipe peut travailler pour toutes les fonctionnalités, car les fonctionnalités sont livrées de manière séquentielle.

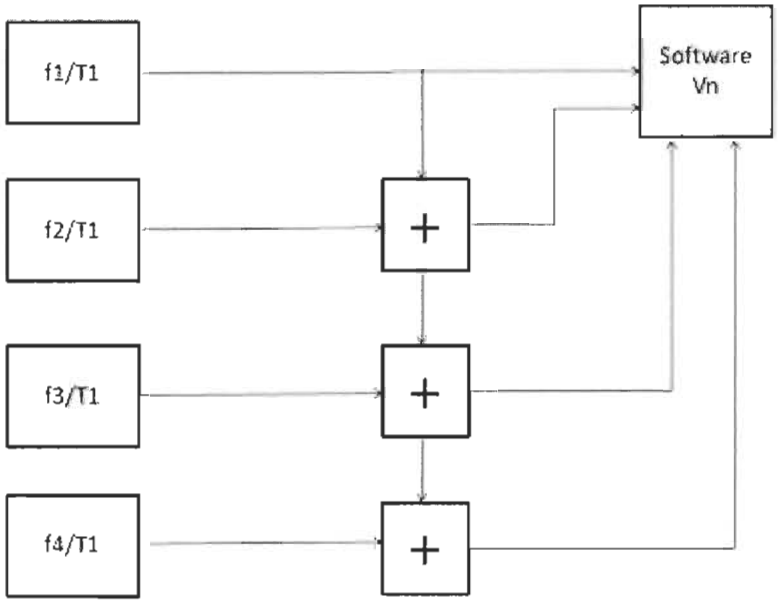

Figure 7 – Le développement purement incrémentiel

Dans un projet purement incrémentiel, chaque nouvelle version logicielle ne reçoit une nouvelle fonctionnalité que lorsqu'elle est terminée. L'idée de backlog est introduite lorsque les incréments sont ajoutés au backlog final. L'intégration continue est également introduite, car nous avons une intégration à mesure que chaque incrément est ajouté à la version du logiciel. La première version du logiciel est par exemple:

V1 = f1 + f2

Ensuite, la deuxième version du logiciel est:

V2 = f1 + f2 + f3

Et la troisième version du logiciel est:

V3 = f1 + f2 + f3 + f4

Dans le cas d'une régression logicielle, la fonctionnalité complète est supprimée, par exemple, la quatrième version logicielle est:

V4 = f1 + f2 + f4

Il n'est pas habituel d'avoir un projet purement incrémentiel ou un projet purement itératif sauf si c'est un cas très particulier. Le processus le plus couramment utilisé pour un projet logiciel complexe est à la fois incrémentiel et itératif.

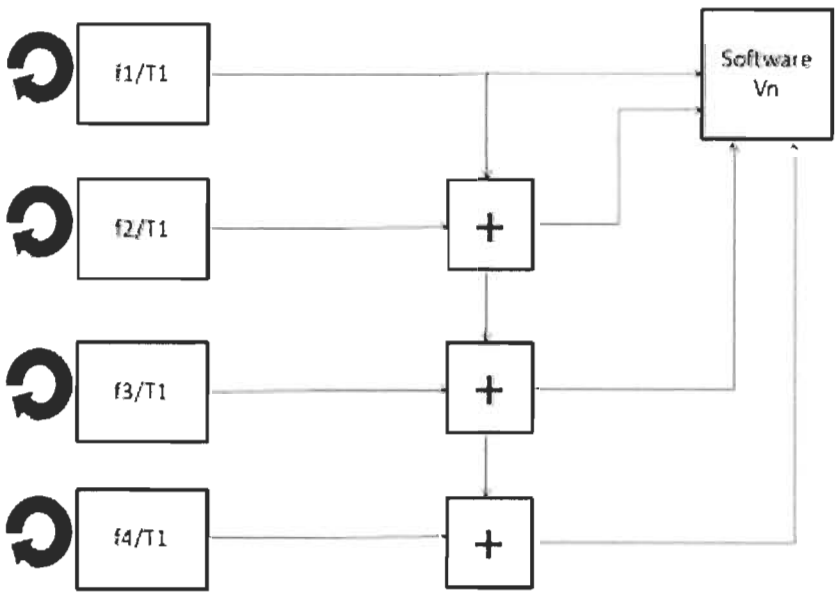

Figure 8 – Le développement itératif et incrémentiel

Dans le développement itératif et incrémentiel, la première version logicielle peut recevoir plusieurs fonctionnalités, chaque fonctionnalité étant finale ou non. Les fonctionnalités sont développées de manière itérative, et elles sont ajoutées

ou supprimées du logiciel ou remplacées par une nouvelle ou une ancienne version (en cas de régression). Une fonctionnalité complète peut également être supprimée de la nouvelle version du logiciel, ce qui donne une totale liberté dans le processus de développement.

Par exemple, imaginez une première version du logiciel comme étant:

V1.0 = f1 + f2

La deuxième version reçoit des mises à jour pour les deux fonctionnalités f1 et f2 (processus itératif) et se transforme en version V1.1.

V1.1 = f1' + f2'

La troisième version reçoit une nouvelle fonctionnalité f3 (processus incrémentiel) et une mise à jour dans la fonctionnalité f2, et cette version peut être V2.0.

V2.0 = f1' + f2'' + f3

Dans cet exemple, le premier chiffre du numéro de version représente les nouvelles fonctionnalités (révision majeure) et le second représente la mise à jour des fonctionnalités (révision mineure). Un troisième chiffre pourrait représenter des corrections de bogues. Chaque entreprise a ses propres règles pour les numéros de révision et les règles de contrôle des révisions sont hors de portée de ce travail.

Cockburn définit séparément le développement incrémentiel et le développement itératif (Cockburn, 2008):

"Le développement incrémentiel est une tactique de mise en scène et de programmation dans laquelle quelques partions du système sont développées en différents instants ou à différents taux et intégrées au fur et à mesure qu'elles sont terminées. La tactique alternative au développement incrémentiel consiste à développer l'ensemble du système qui mène à une intégration "big bang" à la fin.

Le développement s'appelle itératif quand il y a une stratégie de planification de retouches dans laquelle le temps est gardé pour la révision et amélioration de certaines parties du système. Une tactique alternative pour le développement itératif consiste à planifier pour que tout se passe bien à la première fois."

La définition de développement itératif et incrémentiel n'est rien de plus compliqué que d'utiliser à la fois des processus itératifs et incrémentiels dans un projet.

2.3.2. Gestion de projet évolutive (EVO)

Le premier livre que nous pouvons trouver sur la gestion de projet évolutive est "Software Metrics", par Tom Gilb, de 1976 (Larman et al., 2003).

L'Agile Practice Guide déclare que la livraison de valeur évolutive (EVO) est la première Méthode Agile contenant un composant qu'aucune autre méthode n'a: le focus mis sur la livraison de multiples exigences de valeur mesurables aux parties prenantes (Project Management Institute, 2017b).

La gestion de projet évolutive a été officiellement définie par le ministère américain de la Défense en 1994 par le biais de la norme militaire MIL-STD-498. Le concept selon lequel les projets n'ont pas les exigences finales et correctes des utilisateurs à ses débuts a été soutenu. Cette norme influence toujours les normes civiles, comme les normes IEEE (Gilb et al., 2005).

Le processus de gestion de projet évolutive utilise le cycle de «planifier-faire-étudier-agir» et exige la livraison rapide des résultats du projet aux parties prenantes, des diffusions fréquentes aux parties prenantes, de petites incréments, des étapes utiles aux parties prenantes (avantage livrée, valeur expérimentée), sélection et séquencement des étapes en fonction du degré d'avantage pour les parties prenantes, et généralement, mais pas toujours, des étapes à haut bénéfice en premier (en utilisant la détermination des priorités dynamiques) (Gilb et al., 2005). EVO utilise également le feedback incrémentiel, la livraison incrémentielle (Gilb, 2003) et offre une série d'opportunités de prototypage (Gilb, 2005).

La politique de planification des projets évolutifs a des contraintes strictes pour les contrôles financiers, de délais et de valeur. Pour le contrôle financier, aucun cycle de projet ne peut dépasser 2% du budget financier initial total avant de fournir des résultats mesurables et requis aux parties prenantes. Pour le contrôle des délais, aucun cycle de projet ne peut dépasser 2% du temps total du projet (par exemple, une semaine pour un projet d'un an) avant de fournir des résultats mesurables et requis aux parties prenantes. Et pour le contrôle de la

valeur, la prochaine étape devrait toujours être celle qui offre la meilleure valeur aux parties prenantes pour ses coûts (Gilb et al., 2005).

Tom Gilb déclare que le modèle évolutif est recommandé pour les projets imprévisibles, complexes et à taux de changement élevé, et non pour les projets prévisibles, à faible risque, à faible turbulence et à faible compétition (Gilb et al., 2005).

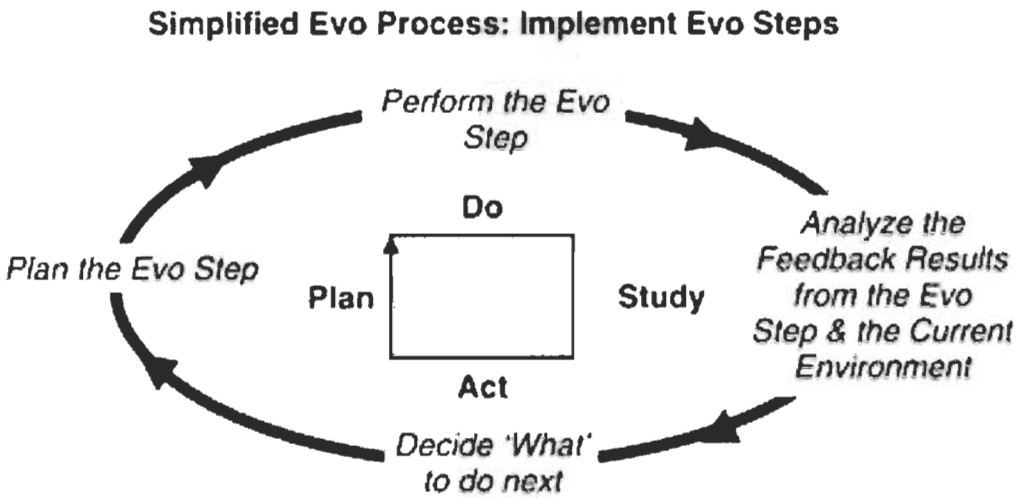

Figure 9 – Processus Evo simplifié: implémentation d'Evo Steps

(Crédits d'image: Gilb et al., 2005)

Les 10 principes de la gestion de projet évolutive peuvent être définis par (Gilb et al., 2005):

1. Le principe du "prochain pas de Capablanca":

Il n'y a qu'un seul coup qui compte vraiment, le suivant.

2. Le principe "Faites d'abord les morceaux juteux":

Faites tout ce qui donne les plus gros gains. Ne laissez pas les autres trucs vous distraire!

3. Le principe "Mieux vaut le diable que vous connaissez":

Les visionnaires qui réussissent commencent là où ils sont, ce qu'ils ont et ce que leurs clients ont.

4. Le principe "vous mangez un éléphant une bouchée à la fois":

Les parties prenantes du système doivent digérer les nouveaux systèmes par petits incréments.

5. Le principe de "cause et effet":

Si vous changez par petites étapes, les causes des effets sont plus claires et plus faciles à corriger.

6. Le principe du "lève-tôt attrape le ver":

Vos clients seront plus heureux avec un flux précoce à long terme de leurs améliorations prioritaires, que des années de promesses, aboutissant à une catastrophe tardive.

7. Le principe de la "frappe tôt, alors que le fer est encore chaud":

Installez rapidement de petites étapes avec les personnes les plus intéressées et motivées.

8. Le principe "Un oiseau dans la main vaut deux dans la brousse":

Votre prochaine étape devrait donner le meilleur résultat que vous pouvez obtenir maintenant.

9. Le principe "aucun plan ne survit au premier contact avec l'ennemi":

Une petite expérience pratique bat beaucoup de réunions de comité.

10. Le principe de "l'architecture adaptative":

Comme vous ne savez pas où et quand vous allez, votre première priorité est de vous équiper pour aller presque n'importe où, n'importe quand.

Selon le processus de gestion de projet évolutif, le projet doit être décomposé en petites ''étapes évolutives'', et Gilb (Gilb et al., 2005) définit une liste de 20 conseils pratiques.

Gilb définit également certains concepts pour définir la gestion de projet évolutive (Gilb, 2005):

• Faites des changements utiles pour vos parties prenantes, tôt et fréquemment;

• Apprendre de l'expérience des systèmes vivantes, ce qui fonctionne vraiment et ce qui ne fonctionne pas;

• Utilisez une spécification numérique de vos principaux objectifs de performance;

• Estimer les impacts numériques des idées de conception sur vos besoins;

• Mesurer l'effet de chaque étape Evo;

• Concentrez-vous à livrer la valeur disponible la plus élevée à chaque étape;

• Ne planifiez pas trop longtemps à l'avance - vous avez des objectifs à long terme, mais vous devez réagir aux réalités à court terme pour les atteindre.

2.3.3. ASD (Adaptive Software Development)

Le développement logiciel adaptatif (Adaptive Software Development) est défini par Highsmith comme consistant en trois composants, le modèle conceptuel adaptatif (Adaptive Conceptual Model), le modèle de développement adaptatif (Adaptive Development Model) et le modèle de gestion adaptative (Adaptive Management Model) (Highsmith, 2000).

Le modèle conceptuel adaptatif (systèmes adaptatifs complexes), définit la mentalité nécessaire pour pouvoir utiliser les modèles de développement et de gestion (Highsmith, 2000).

Le modèle de développement adaptatif (spéculer, collaborer, apprendre), établit la grande vitesse par l'utilisation de l'itération, de la concurrence, du feedback et de la collaboration (Highsmith, 2000).

Le focus du modèle de gestion adaptative (leadership et collaboration) vise à créer la culture adaptative et à identifier les pratiques adaptatives, à impliquer des groupes de travail distribués et à gérer les changements élevés, la collaboration et la gestion des résultats. L'idée est d'adapter le modèle de gestion à des environnements où l'équilibre n'est pas la condition normale et le changement est la norme, comme les environnements à grande vitesse et à changement élevé. Highsmith dit également que le focus sur le modèle de gestion adaptative se concentre sur les aspects culturels et structurels de la collaboration nécessaires pour créer un ordre émergent à plus globalement et étendre le développement adaptatif jusqu'à des projets complexes plus gros (Highsmith, 2000). La collaboration fonctionne bien avec les équipes interfonctionnelles et Highsmith recommande des réunions facilitées ou des sessions JAD (Joint Application Development) pour rassembler des groupes interfonctionnels afin de construire des relations de collaboration capables de produire des livrables de haute qualité dans un court laps de temps (Highsmith, 2000). Selon Highsmith, les meilleures équipes sont petites; une équipe centrale devrait comprendre moins de dix membres.

Highsmith définit également une opposition entre les modèles adaptatifs et les modèles traditionnels en disant que si ce dernier est une approche qui dépend d'un "ordre imposé", le premier a une approche qui passe par un "ordre émergent". Les modèles traditionnels reposent sur la croyance et les actions qui dépendent de la capacité de prédire l'avenir et de le réaliser (ordre imposé). Le

modèle adaptatif a la propriété de "l'émergence" qui est la capacité de produire des résultats par l'interaction spontanée d'agents auto-organisateurs. Cette idée est bien illustrée dans la *Figure 10* (Highsmith, 2000).

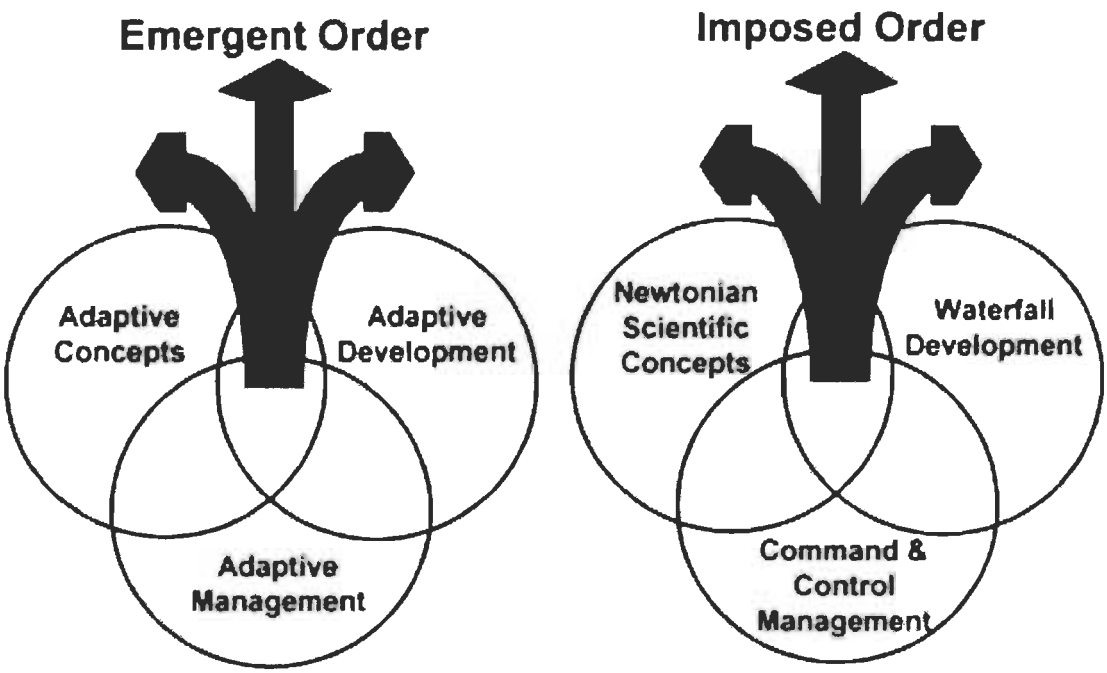

Figure 10 – Les principales différences entre les modèles adaptatifs et traditionnels

(Crédits d'image: Highsmith, 2000)

Highsmith dit que l'ASD est venu en réponse au fait que RAD (Rapid Application Development - §2.3.4. RAD (Rapid Application Development)) a des problèmes avec des taux de changement élevés. Il compare les approches ASD, RAD, cascade et evolutive dans les axes de changement faible/élevé et de vitesse faible/élevée. Alors que l'approche en cascade fonctionne bien avec une

faible vitesse et de faibles taux de changement, l'approche évolutive traite des taux de changement élevés au détriment de la vitesse (basse vitesse), l'approche RAD à grande vitesse traite généralement de petits volumes de changements et l'ASD est orienté à la fois à haute vitesse et un volume élevé de changements (Highsmith, 2000).

Tableau 2 - Matrice du cycle de vie

(Crédits d'image: Highsmith, 2000)

	Low Speed	High Speed
Low Change	Waterfall	RAD
High Change	Evolutionary	Adaptive

ASD dispose de techniques de planification adaptative qui peuvent être définies en sept "étapes de planification de cycle", à savoir: mener la phase de démarrage du projet, déterminer le timebox du projet, déterminer le nombre optimal de cycles et le timebox pour chacun, écrire un objectif pour chaque cycle, attribuer les composants primaires aux cycles, attribuer la technologie et les

composants de support aux cycles et développer une liste de tâches de projet (Highsmith, 2000).

ASD traite différemment le problème de la "passage à grande échelle" par rapport aux projets traditionnels. Dans les projets traditionnels, la complexité et la taille croissantes sont gérées en augmentant la rigueur des processus. Dans les projets adaptatifs, la complexité et la taille croissantes sont gérées en augmentant la rigueur appliquée aux flux d'informations. (Highsmith, 2000).

Le développement adaptatif, avec XP, Scrum et Crystal, sont principalement basés sur les meilleures pratiques qui existent depuis de nombreuses années, telles que le développement itératif, l'intégration continue, les user stories et bien d'autres (Kroll et al., 2003).

ASD utilise tous les niveaux de test, test unitaire, test d'intégration, test système et test d'acceptation (Abrahamsson et al., 2003).

2.3.4. RAD (Rapid Application Development)

James Martin a introduit le développement rapide d'applications en 1991, et il l'a décrit comme "un processus de développement logiciel qui met l'accent sur une approche itérative de l'ensemble de la phase de construction et gère les prototypes en tant que citoyens de première classe du processus de mise en œuvre" (Novák, 2011).

Le développement rapide d'applications est un groupe de méthodologies de développement logiciel qui utilise l'itération et des prototypes - Scrum, Extreme Programming (XP), Lean Software Development (LD) ou Joint Application Development (JAD) sont inclus dans ce groupe (Novák, 2011).

Dans le développement rapide d'applications, le prototypage est plus important que la planification - les prototypes sont développés de manière itérative.

Alors que l'itération a déjà été discutée dans le présent travail lorsque nous avons parlé de développement itératif et incrémentiel, il est maintenant temps de parler de prototypage.

Le prototypage est la production d'un modèle fonctionnel d'une ou de plusieurs parties du système pour prouver certains concepts issus des exigences. Il peut être utilisé pour les tests ou au moins pour la visualisation.

Certains composants utilisés dans RAD sont (mais sans s'y limiter): petites équipes, développement de courte durée, salles blanches (clean-rooms), timeboxing, prototypage incrémentiel et outils de développement rapide (Beynon-Davies et al., 1999).

Petites équipes RAD:

RAD utilise généralement de petites équipes de développement de quatre à huit personnes, entre développeurs et testeurs, qui sont habilitées à prendre des décisions, et qualifiées à la fois socialement et en termes d'affaires,

autrement dit, elles ont des compétences à la fois sociales et techniques (Beynon-Davies et al., 1999).

Développement de courte durée RAD:

Les projets RAD typiques sont de petite envergure et de courte durée - une durée de deux à six mois est considérée comme normale (Beynon-Davies et al., 1999).

Salles blanches RAD:

Les projets de RAD sont connus pour être menés dans des salles blanches, "des lieux sans interruptions de travail quotidiennes et pleins des installations de soutien nécessaires telles que des tableaux à feuilles mobiles, des post-its, du café, des ordinateurs, etc." L'accent est mis sur la résolution de problèmes (Beynon-Davies et al., 1999).

Timeboxing RAD:

Les projets RAD ont une priorité élevée sur les délais, il est donc normal de réduire les exigences pour respecter le timebox lorsque le calendrier risque de retarder, au lieu de reporter la date de livraison (Beynon-Davies et al., 1999).

Prototypage incrémentiel RAD:

Nous avons le prototypage incrémentiel comme l'un des composants les plus importants d'un projet RAD. Des prototypes sont construits par l'équipe pour démontrer aux utilisateurs la vision d'équipe du système. Après la démonstration, les développeurs et les utilisateurs discutent des améliorations et des corrections sur le système et un nouveau cycle de construction/démonstration de prototype démarre. Ce cycle se répète jusqu'à ce que les utilisateurs soient satisfaits du système (Beynon-Davies et al., 1999).

Outils de développement rapide RAD:

Les projets RAD nécessitent une bonne prise en charge d'outils, tels que les langages de quatrième génération, les constructeurs d'interfaces utilisateur graphiques (GUI – Graphical User Interface), les systèmes de gestion de base de données (SGBD) et les outils d'ingénierie logicielle assistée par ordinateur (CASE – Computer Aided Software Engineering). Ces outils permettent des changements rapides dans les prototypes (Beynon-Davies et al., 1999).

Certaines des caractéristiques des projets RAD sont (McFarlane, 2004):

- Moins de temps, même effet: la meilleure solution est la plus rapide pour obtenir les résultats.

- Prototypes visuels: les humains étant majoritairement des utilisateurs visuels, les projets RAD doivent pouvoir créer rapidement des prototypes aussi visuels que possible.

- Solutions verticales: elles sont définies comme des produits à finalité étroite, elles n'ont pas besoin d'être flexibles à l'origine.

- Logiciel COTS par rapport à fait à la maison: l'utilisation du code déjà produit par d'autres permet d'économiser du temps et des efforts nécessaires dans les projets RAD.

- Tests déstructurés: ceci est basé sur l'argument selon lequel les petits morceaux de code déjà testé sont moins sujets aux erreurs lorsqu'ils sont utilisés dans un code plus grand, simplifiant les tâches de test.

En parlant de simplifier les tâches de test, en raison de la nature rapide du RAD, il est prévu d'utiliser des tests automatisés en son sein. Des tests manuels sont également utilisés, car c'est le meilleur moyen de tester la fonctionnalité d'affaires de votre application, en créant une liste de cas de test manuels qui peuvent être refaites avant chaque déploiement (Novák, 2011).

Novák recommande Microsoft SharePoint et Lync comme plates-formes de collaboration (Novák, 2011).

La fréquence du feedback doit être correctement dimensionnée. S'il est trop faible et que les utilisateurs ne peuvent voir le nouveau système qu'aux phases finales, il peut être trop long et coûteux à réparer. S'il est trop élevé, le temps

passé à montrer le système et à obtenir des commentaires demande des ressources qui devraient être investies dans le développement (Novák, 2011).

2.3.5. RUP (Rational Unified Process)

Le Rational Unified Process (RUP) est un processus hautement itératif où, à chaque itération, certaines exigences, analyses, conception, implémentation et tests sont effectués. Comme le montre la *Figure 11*, chaque itération ajoute au travail effectué dans les itérations précédentes pour se rapprocher du produit final (Kroll et al., 2003).

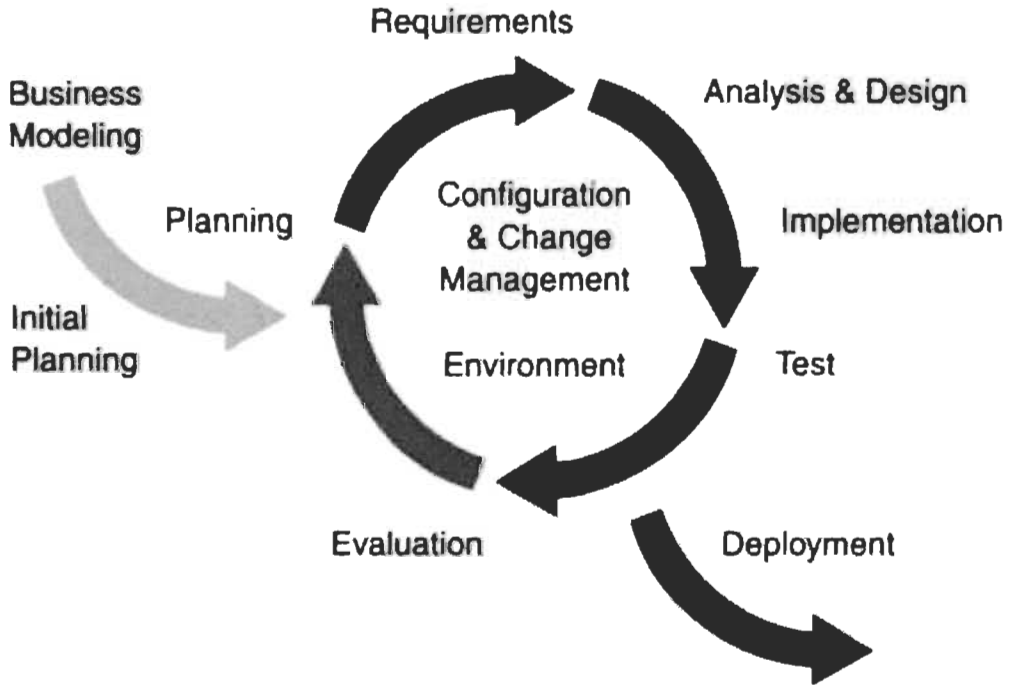

Figure 11 – Développement itératif dans le RUP

(Crédits d'image: Kroll et al., 2003)

Le processus unifié rationnel est basé sur deux dimensions (*Figure 12*): l'aspect dynamique (horizontal) est lié aux cycles, phases, itérations et jalons; l'aspect statique (vertical) est lié aux activités, disciplines, artefacts et rôles (Kroll et al., 2003). RUP utilise une approche itérative (itérer = répéter), c'est-à-dire une séquence d'étapes ou d'itérations incrémentielles (itérative et incrémentiele). Chaque itération comprend certaines ou la plupart des disciplines de développement (exigences, analyse, conception, implémentation, etc.). Chaque itération est un cycle de développement logiciel complet exécuté en très peu de temps.

Selon Kroll, les tests automatisés sont essentiels au développement itératif, car il est très difficile de faire un développement itératif sans un bon support pour les tests automatisés, entre autres (Kroll et al., 2003). Les tests manuels et automatisés doivent être organisés en suites de tests. Les tests unitaires doivent être effectués au fur et à mesure de la mise en œuvre des composants, pour être sûr qu'ils correspondent aux spécifications.

En parlant des membres de l'équipe travaillant ensemble, Kroll dit que les équipes doivent être dotées des outils nécessaires pour une collaboration efficace entre les fonctions (Kroll et al., 2003).

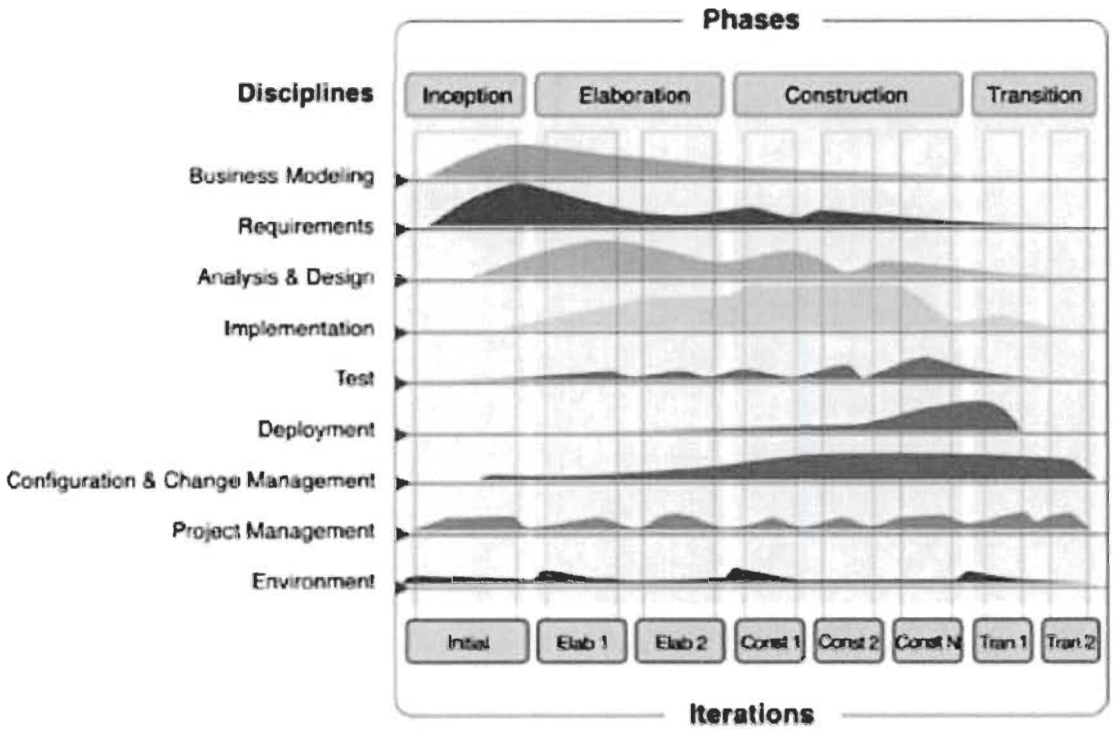

Figure 12 – Les deux dimensions du RUP

(Crédits d'image: Kroll et al., 2003)

Rational Software a activement promu le développement itératif en mettant fortement l'accent sur les logiciels exécutables plutôt que sur les sous-produits (Kroll et al., 2003). Le processus unifié Rational est un cadre de processus qui vous permet de produire des configurations de processus personnalisées qui sont itératives et axées sur les risques, avec des tests et intégration continus.

Kroll recommande le timeboxing, car c'est une technique top-down qui essaie de limiter la réalisation d'un objectif spécifique dans un petit intervalle de temps. Cela fournit une structure pour se concentrer sur les objectifs de mission les plus importants et forcer les compromis d'ingénierie.

Avec RUP, vous avez des équipes interfonctionnelles, où chaque membre de l'équipe se sent responsable de l'application et de l'équipe qui progresse (Kroll et al., 2003). Avoir une réunion quotidienne rapide au cours de laquelle l'équipe discute la situation et décide sur quoi se concentrer ensuite peut également faciliter le renforcement de l'équipe. L'exécution de builds fréquents garantit des tests d'intégration fréquents, qui fournissent des commentaires sur le code récent qui a été écrit depuis la dernière build.

Parlant de la taille de l'équipe, Kroll dit que le nombre de développeurs devrait être limité au début du projet. Vous voulez éviter d'avoir trop de personnes dans votre projet au début. Vous commencez généralement avec une petite équipe dans Inception, augmentez un peu dans Élaboration, puis ajoutez plus de développeurs et de testeurs à l'équipe pendant la Construction.

L'un des principaux avantages du développement itératif est qu'il vous permet de lancer des tests beaucoup plus tôt que dans le développement en cascade (Kroll et al., 2003). Déjà dans la première phase, Inception, vous testez certaines des fonctionnalités capturées dans les prototypes, vous fournissant de précieux commentaires sur les principaux cas d'utilisation.

Kroll considère que certaines bonnes pratiques des développeurs doivent être utilisées dans RUP, telles que tester fréquemment, intégrer en continu, effectuer une analyse d'exécution, tester d'abord, refactoriser votre code, utiliser des patrons de conception (design patterns), des mécanismes architecturaux et d'autres actifs réutilisables, garder votre conception simple et la programmation en binôme (pair programming).

2.3.6. DSDM (Dynamic Systems Development Method)

DSDM, méthode de développement de systèmes dynamiques, lancée en 1994, est considérée comme la première méthode de développement logiciel vraiment Agile et trouve son origine dans le développement rapide d'applications (RAD §2.3.4. RAD (Rapid Application Development)) (Abrahamsson et al., 2003).

Le DSDM a une livraison axée sur les contraintes, il prend le coût, le temps et la qualité comme contraintes et définit la portée pour répondre à ces contraintes (Project Management Institute, 2017b).

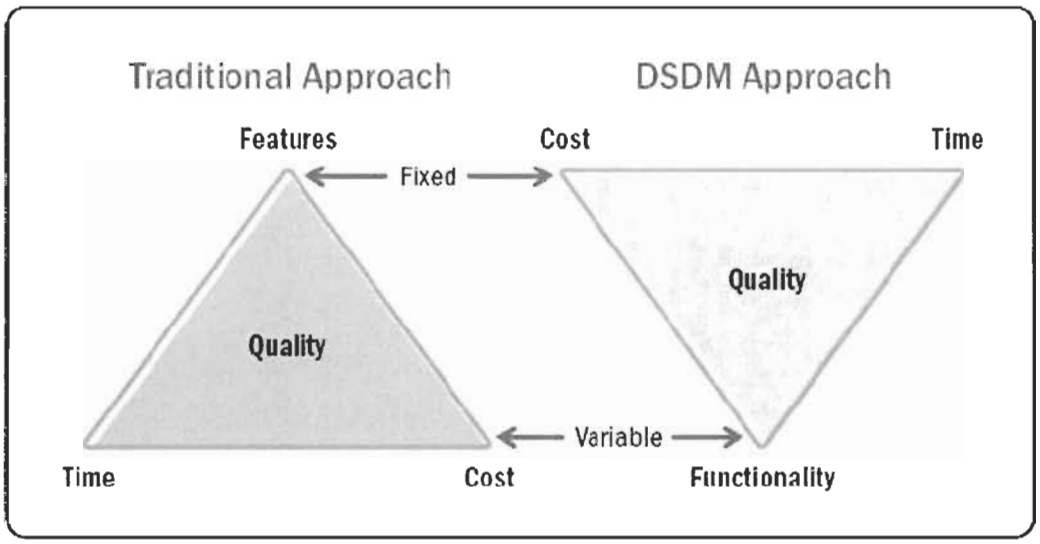

Figure 13 – Approche DSDM pour l'agilité basée sur les contraintes

(Crédits d'image: Project Management Institute, 2017b)

La philosophie DSDM s'appuie sur huit principes (Agile Business Consortium, 2019):

1. Se concentrer sur les besoins d'affaires

Afin de respecter ce principe, les équipes DSDM vont:

- Comprendre les véritables priorités d'affaires
- Établir un cas d'affaires valide
- Assurer un parrainage et un engagement d'affaires continus
- Garantie de livraison de la sous-section minimale utilisable

2. Livrez à temps

Pour respecter ce principe, les équipes DSDM doivent:

- Mettre le travail sur un timebox

- Se concentrer sur les priorités d'affaires

- Toujours respecter les délais

- Renforcez la confiance grâce à une livraison prévisible

3. Collaborez

Pour respecter ce principe, les équipes DSDM doivent:

- Impliquer les bonnes parties prenantes, au bon moment, tout au long du projet

- Encourager la participation proactive des représentants d'affaires

- Veiller à ce que tous les membres de l'équipe soient autorisés à prendre des décisions au nom de ceux qu'ils représentent

- Construire une culture d'équipe unique

4. Ne compromettez jamais la qualité

Pour respecter ce principe, les équipes DSDM doivent:

- Convenir du niveau de qualité dès le départ, avant le début du développement

- Veillez à ce que la qualité ne devienne pas une variable

- Testez tôt, testez en continu et testez au niveau approprié

- Construire en qualité par un examen constant

- Concevoir et documenter de manière appropriée

5. Construire de façon incrémentielle à partir de fondations solides

Pour respecter ce principe, les équipes DSDM doivent:

- Effectuez une analyse appropriée et assez de design à l'avance pour créer des bases solides

- Réévaluer de façon formelle les priorités et réévaluer de manière informelle la viabilité du projet en cours avec chaque incrément livré

6. Développer itérativement

Pour respecter ce principe, les équipes DSDM doivent:

- Construire les feedbacks d'affaires à chaque itération

- Reconnaître que la plupart des détails devraient apparaître plus tard que plus tôt

- Adoptez le changement - la bonne solution n'évolue pas sans elle

- Utiliser le développement itératif pour encourager la créativité, l'expérimentation et l'apprentissage

- Le changement est inévitable; DSDM permet le changement et exploite ses avantages

7. Communiquez de façon continue et claire

Pour respecter ce principe, les équipes DSDM doivent:

- Encourager la communication informelle en face à face à tous les niveaux

- Organiser des séances quotidiennes de stand-up en équipe

- Utiliser des ateliers, avec un facilitateur le cas échéant

- Utiliser des pratiques de communication visuelle telles que la modélisation et le prototypage

- Démontrer la solution évolutive tôt et souvent

- Gardez la documentation allégée et en temps opportun

- Gérer les attentes de la partie prenante à tous les niveaux tout au long du projet

- Toujours viser l'honnêteté et la transparence dans toute communication

8. Démontrer le contrôle

Afin de respecter ce principe, les équipes DSDM, en particulier le chef de projet et le chef d'équipe, doivent:

- Rendre les plans et les progrès visibles pour tous

- Mesurer les progrès en mettant l'accent sur la livraison des produits plutôt que sur les activités terminées

- Gérer de manière proactive

- Évaluer la viabilité continue du projet en fonction des objectifs d'affaires

- Utiliser un niveau de formalité approprié pour le suivi et les rapports

Les principales pratiques utilisées dans DSDM sont (Agile Business Consortium, 2019):

- Ateliers facilités: type de réunion spécialisée qui a des livrables objectifs clairs, un ensemble de personnes capables de fournir le résultat, et un facilitateur d'atelier pour permettre la réalisation effective de l'objectif.

- Priorisation MoSCoW: technique utilisée pour aider à comprendre et gérer les priorités. Les lettres représentent: Must Have, Should Have, Could Have and Won't Have this time (doit être fait, devrait être fait, pourrait être fait et ne sera pas fait cette fois), utilisé pour définir les priorités des exigences.

- Développement itératif: déjà défini dans la section 2.3.1. Développement itératif et incrémentiel.

- Modélisation: utilisation de modèles, prototypes et maquettes pour établir les exigences, confirmer les attentes et tester la faisabilité des objectifs.

- Timeboxing: période de temps fixe, à la fin de laquelle un objectif a été atteint.

Le timebox structurée DSDM comporte trois étapes principales, comme la *Figure 14*:

- Enquête (Investigation)

- Raffinement (Refinement)

- Consolidation

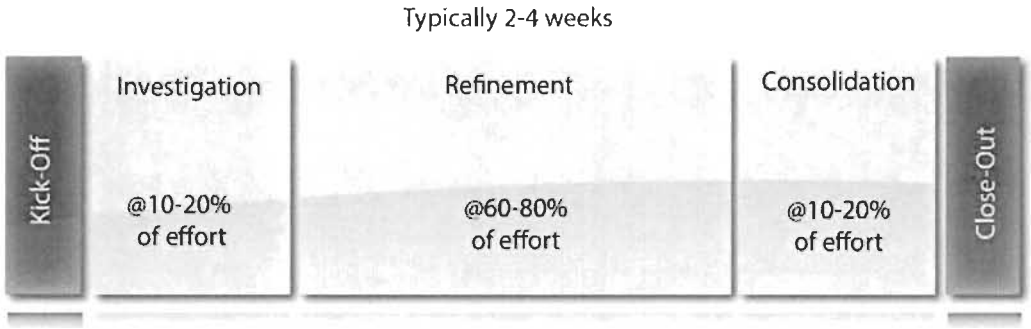

Figure 14 – DSDM Timebox structuré

(Crédits d'image: Agile Business Consortium, 2019)

Les tests manuels et automatisés sont recommandés pour le DSDM (Agile Business Consortium, 2019). Dans le DSDM, le backlog du produit est appelé Liste des Exigences Prioritaires (Prioritized Requirements List). DSDM utilise tous les niveaux de test, test unitaire, test d'intégration, test système et test d'acceptation (Abrahamsson et al., 2003).

Les informations de la rétrospective et du review formel du produit aident à façonner les plans pour les incréments futurs et peuvent être utilisées pour faciliter l'apprentissage entre les projets d'un portefeuille (Agile Business Consortium, 2019).

Avec DSDM, où l'accent est mis sur la livraison fréquente de produits, les décisions erronées peuvent être corrigées, tandis que le cycle de livraison est court et que les utilisateurs peuvent fournir des feedbacks précis (Abrahamsson et al., 2002).

DSDM a été appliqué aux petits et grands projets. Les grands projets sont divisés en composants qui peuvent être développés en petites équipes (Abrahamsson et al., 2002).

2.3.7. Scrum

Scrum ou Scrummage est un terme qui vient du football de rugby qui signifie un type de reprise de jeu où les joueurs sont réunis étroitement pour tenter de prendre possession du ballon.

Scrum dans le contexte du développement logiciel est un framework Agile définissant comment un produit logiciel doit être développé. Le mot Scrum a été utilisé pour la première fois dans un article de Harvard Business Review de 1986, "The New New Product Development Game" (Takeuchi et al., 1986), par Hirotaka Takeuchi et Ikujiro Nonaka.

Dans cet article, Takeuchi et Nonaka comparent la course de relais, où un seul membre de l'équipe travaille seul à la fois, au rugby, où les membres de l'équipe travaillent généralement en même temps, en particulier dans la mêlée avec laquelle ils jouent très près les uns des autres avec le même objectif.

La méthode classique en cascade est un exemple de la façon dont le processus est similaire à une course de relais. Le processus est séquentiel, les phases sont bien divisées, sont différentes les unes des autres et la phase suivante ne démarre que lorsque toutes les exigences de la phase précédente sont remplies.

Selon le Scrum Guide (Schwaber et al., 2016), Scrum est "un cadre dans lequel les gens peuvent résoudre des problèmes adaptatifs complexes, tout en fournissant de manière productive et créative des produits de la plus haute valeur possible."

De manière pratique, le framework Scrum est défini généralement par:

- Piliers Scrum: transparence, inspection et adaptation.

- Valeurs Scrum: engagement, courage, focus, ouverture et respect.

- Scrum Team: Propriétaire du Produit (Product Owner), l'équipe de Développement et Scrum Master.

- Événements Scrum: Sprint, Planification du Sprint, Daily Scrum, Revue de Sprint et Rétrospective de Sprint.

- Artefacts de Scrum: Backlog Produit, Backlog Sprint et Incrément.

Les équipes Scrum sont auto-organisées et interfonctionnelles. L'équipe est libre de décider de la meilleure façon de travailler et ses membres ont toutes les connaissances et les capacités pour accomplir les tâches.

Le Scrum Master est responsable de s'assurer que l'équipe Scrum adhère à la théorie, aux pratiques et aux règles Scrum. Le Scrum Master agit comme un facilitateur en supprimant les obstacles à la progression de l'équipe de développement et en facilitant les événements Scrum comme demandé ou nécessaire. Le Scrum Master assure également la collaboration entre les membres de l'équipe Scrum.

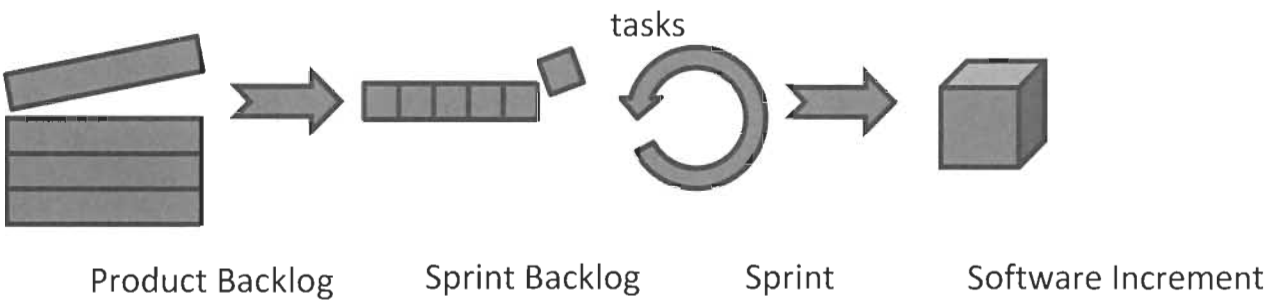

Figure 15 – Le processus Scrum

Le processus Scrum divise le Backlog de produit en Backlog de sprint à développer pendant le sprint (généralement 30 jours), fournissant un incrément de travail du logiciel (*Figure 11*). Le processus Scrum est optimisé pour fournir un feedback pendant les itérations. L'utilisation de prototypes est importante car chaque itération fournit des incréments logiciels de produit non final - le prototype peut avoir à la fois produits déjà développés et des maquettes - la partie "à développer" peut ensuite être visualisée par les parties intéressées et l'équipe Scrum peut avoir des commentaires pour ce qui a été fait et ce qui doit être fait en même temps.

Le Sprint commence avec le "Sprint Planning" lorsque l'équipe reçoit les "Epics" et joue un "Planning Poker", où chaque "User Story" est dimensionnée en points concernant la taille de l'effort du développement, les tests et la documentation. Chaque Epic est une collection de "User Stories". L'User Story

est généralement définie en utilisant le format suivant: "En tant que <rôle utilisateur>, je veux <faire quelque chose> pour que <résultat attendu>".

Chaque jour de la période de sprint de 30 jours a une réunion quotidienne de 15 minutes (mêlée quotidienne ou Daily Scrum) où l'équipe de développement synchronise les activités et crée des plans pour les prochaines 24 heures. Lors de cette rencontre, chaque membre de l'équipe présente une réponse aux 3 questions:

1. Qu'est-ce que j'ai fait pour aider l'équipe à atteindre l'objectif de sprint?
2. Que vais-je faire pour aider l'équipe à atteindre l'objectif de sprint?
3. Est-ce que je vois un obstacle qui m'empêche, moi ou l'équipe, d'atteindre l'objectif de sprint?

Ces trois questions créent une sorte de mentalité, et chaque membre de l'équipe doit être prêt à y répondre tous les jours. Cet état d'esprit oblige le membre de l'équipe à se vérifier tout le temps qu'il travaille sur le produit. Ces trois questions sont une sorte de liste de contrôle à exécuter quotidiennement au cours de la réunion quotidienne, et elle peut être étendue pour se conformer aux exigences du produit ou aux exigences de l'entreprise.

Suivi des progrès

Le suivi des progrès est une très bonne pratique, et le tableau burn-down est un moyen de bien comprendre l'état du projet.

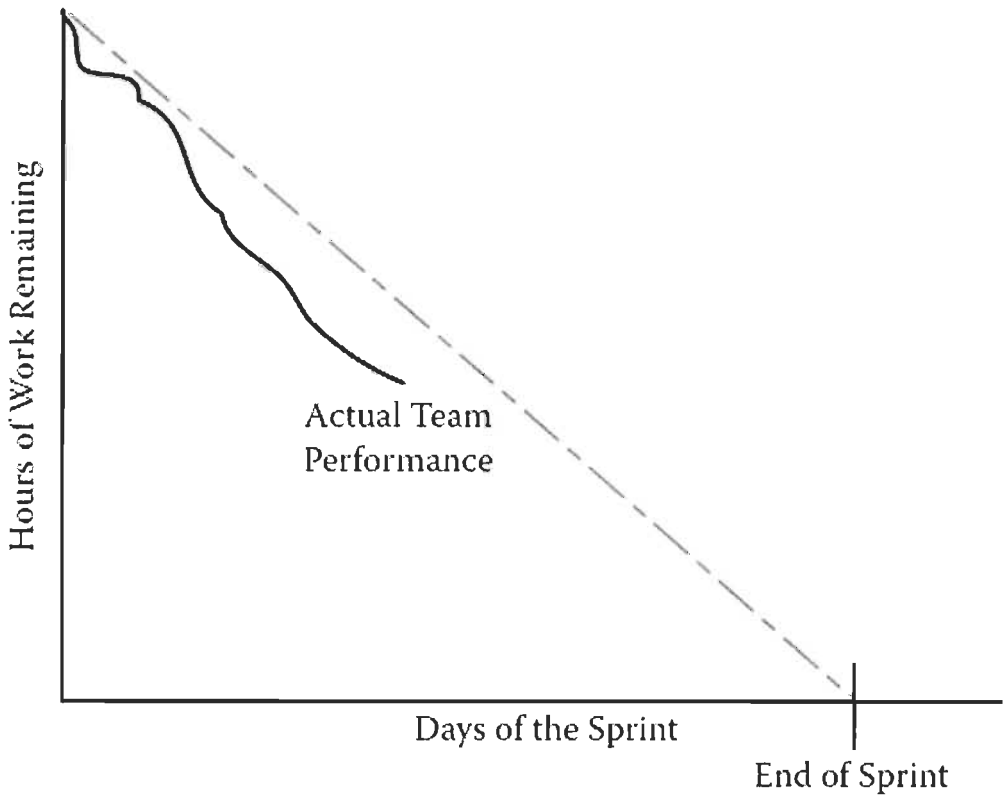

Figure 16 – Tableau burn-down

(Crédits d'image: Pries et al., 2011)

La revue de sprint est le moment pour l'équipe d'exposer les résultats du sprint, et une liste de contrôle est généralement exécutée après que les preuves des résultats du sprint ont été montrées. Cette liste de contrôle est utilisée pour s'assurer que toutes les normes de l'entreprise ont été respectées, comme la vérification de la couverture du test, le test de régression, si les tests ont été effectués sur une base de données à jour, sur la branche master et après la fusion du code dans la branche master.

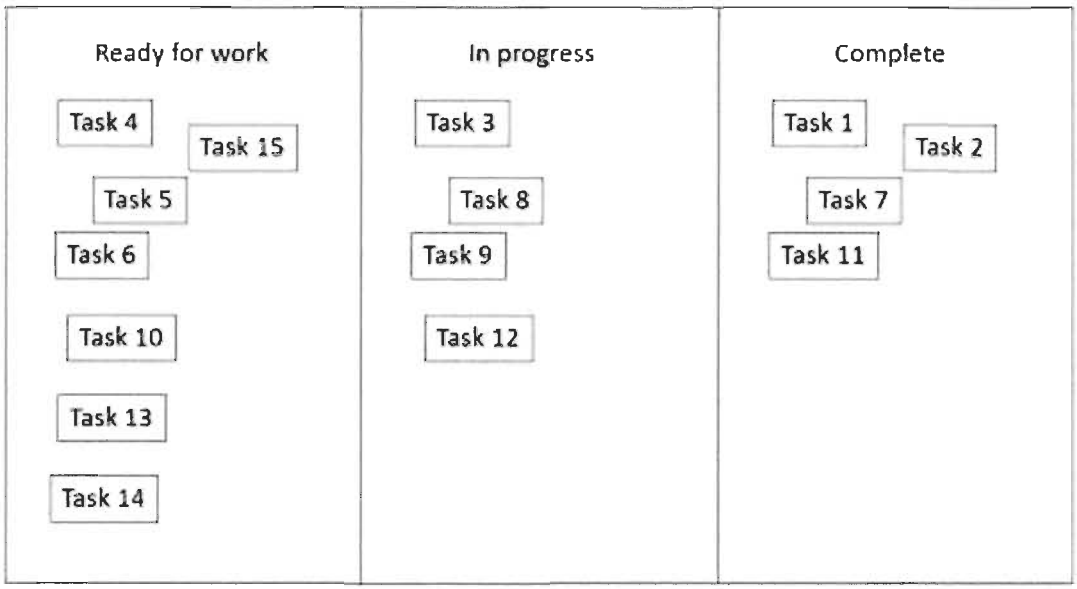

Figure 17 – Système Kanban

Le système Kanban a été développé par Taiichi Ohno, inspiré des systèmes d'inventaire just-in-time, comme moyen de planifier le contrôle des stocks et le réapprovisionnement (Project Management Institute, 2017b). Des tableaux Kanban physiques avec des cartes permettent et favorisent la visualisation et le flux du travail à travers le système pour que tout le monde puisse le voir (Pries et al., 2011). Comme nous pouvons le voir sur la *Figure 17*, le travail à effectuer est divisé en tâches simples et placé du côté "Prêt pour le travail" (Ready for work). Ensuite, les membres de l'équipe prennent une tâche à accomplir et la laissent dans la partie "En cour" (In progress). Lorsqu'une tâche est terminée, son autocollant correspondant est déplacé du côté "Terminé" (Complete) et le

membre de l'équipe est prêt à prendre une autre tâche du côté ''Prêt pour le travail''. Cela continue jusqu'à ce qu'il n'y ait plus aucune tâche à faire.

Le système Kanban fonctionne bien avec la nature itérative Scrum. Comme le backlog de sprint est bien divisé en petites tâches à effectuer pendant le sprint, le tableau Kanban permet d'avoir une vue globale de l'ensernble des tâches de sprint.

Comme l'approche Scrum consiste à fournir en permanence un produit réalisable et vendable au client (Pries et al., 2010), il est important d'avoir recours à la stratégie d'intégration continue. Les tests automatisés permettent d'économiser du temps et des efforts et d'accélérer le processus d'itération. Scrum utilise les niveaux de test suivants: test unitaire, test d'intégration et test système (Abrahamsson et al., 2003).

2.3.8. XP (Extreme Programming)

XP (Extreme Programming) est un moyen léger, efficace, à faible risque, flexible, prévisible, scientifique et amusant de développer des logiciels. Elle se distingue des autres méthodologies par (Beck et al., 2004):

• Feedback précoce, concrète et continue de cycles courts.

• Approche de planification incrémentale, qui aboutit rapidement à un plan global qui devrait évoluer tout au long de la vie du projet.

• Capacité à planifier de manière flexible la mise en œuvre de fonctionnalités, répondant aux besoins changeants de l'entreprise.

• Dépendance à des tests automatisés rédigés par les programmeurs et les clients pour suivre la progression du développement, permettre au système d'évoluer et détecter les défauts tôt.

• Dépendance à la communication orale, aux tests et au code source pour communiquer la structure et l'intention du système.

• Dépendance à un processus de conception évolutif qui dure aussi longtemps que le système dure.

• Dépendance à l'étroite collaboration de programmeurs aux compétences ordinaires.

• Dépendance à des pratiques qui fonctionnent à la fois avec l'instinct à court terme des programmeurs et les intérêts à long terme du projet.

D'autres pratiques de programmation extrême sont (Beck et al., 2004):

• Tests précoces, fréquents et automatisés.

• Conception incrémentale.

• Déploiement quotidien.

• Implication du client.

• Intégration continue.

• Cycles de développement courts.

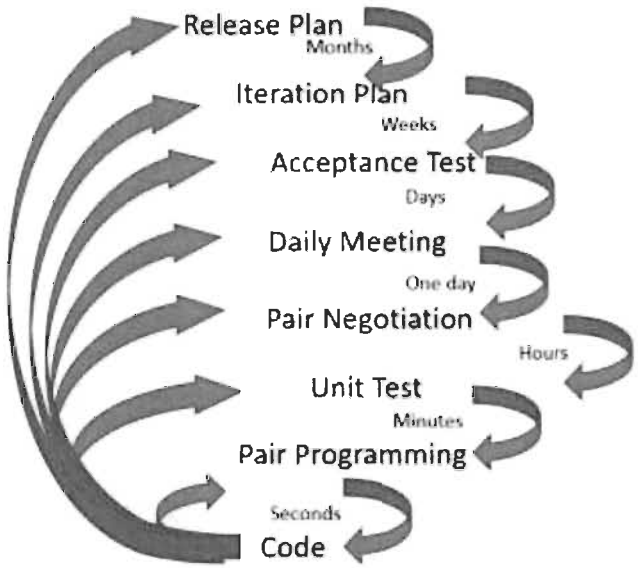

Figure 18 – Feedback d'Extreme programming

En intégrant les tests automatisés dans la boucle interne de la programmation, XP tente de corriger les défauts plus tôt et à moindre coût (Beck et al., 2004). Cela donne aux équipes XP une chance de développer à moindre coût des logiciels avec très peu de défauts par rapport aux normes de leurs contemporains. Bien que la plupart des tests soient automatisés, il reste nécessaire de procéder à des tests manuels, du moins aux stades antérieurs où les tests automatisés sont encore en cours de développement. Comme de nouveaux morceaux de code sont ajoutés à chaque itération, les tests de régression nécessaires pour prouver que l'ancienne partie logicielle est toujours fonctionnelle sont normalement des tests automatisés.

La programmation en binôme (Pair Programming) est habituelle dans Extreme Programming, où deux programmeurs partagent le même ordinateur

pour coder - des tests unitaires sont développés au cours de cette tâche. Ils construisent ensemble le même code de manière collaborative. Selon Beck (Beck et al., 2004), les programmeurs en binôme:

- Gardent les uns les autres sur la tâche.

- Réfléchissent aux améliorations du système.

- Clarifie les idées.

- Prennent l'initiative lorsque leur partenaire est coincé, réduisant ainsi la frustration.

- Tiennent mutuellement responsables des pratiques de l'équipe.

2.3.9. Crystal

Alistair Cockburn définit les méthodologies Crystal comme une famille de méthodologies, ainsi que des principes pour les ajuster (Cockburn, 2005, préface). Alors que le nom de famille est "Crystal", chaque membre de la famille a son nom propre comme "Clear", "Yellow", "Orange" ou "Red", correspondant à la taille du projet. Pour chacune des couleurs, un degré de criticité est ajouté sous forme de lettres, correspondant au niveau des dommages causés par des défauts non détectés, à savoir C (perte de confort), D (perte d'argent discrétionnaire), E (perte d'argent essentiel) et L (perte de vie) (Cockburn, 2005).

En parlant de Méthodes Agiles, Cockburn dit que la Méthode Agile doit être sélectionnée en fonction du nombre de personnes devant être coordonnées et du degré de dommages qui pourraient être causés par un dysfonctionnement du

système (Cockburn, 2005). Dans un autre article, il compare trois des Méthodes Agiles, la famille Crystal, XP et l'ASD (§ 2.3.3. ASD (Adaptive Software Development)), et dit que même si Crystal est optimisé pour la productivité et la tolérance, XP optimise la productivité en réduisant la tolérance, et l'ASD est très ciblé situations de projet instables (Cockburn, 2000).

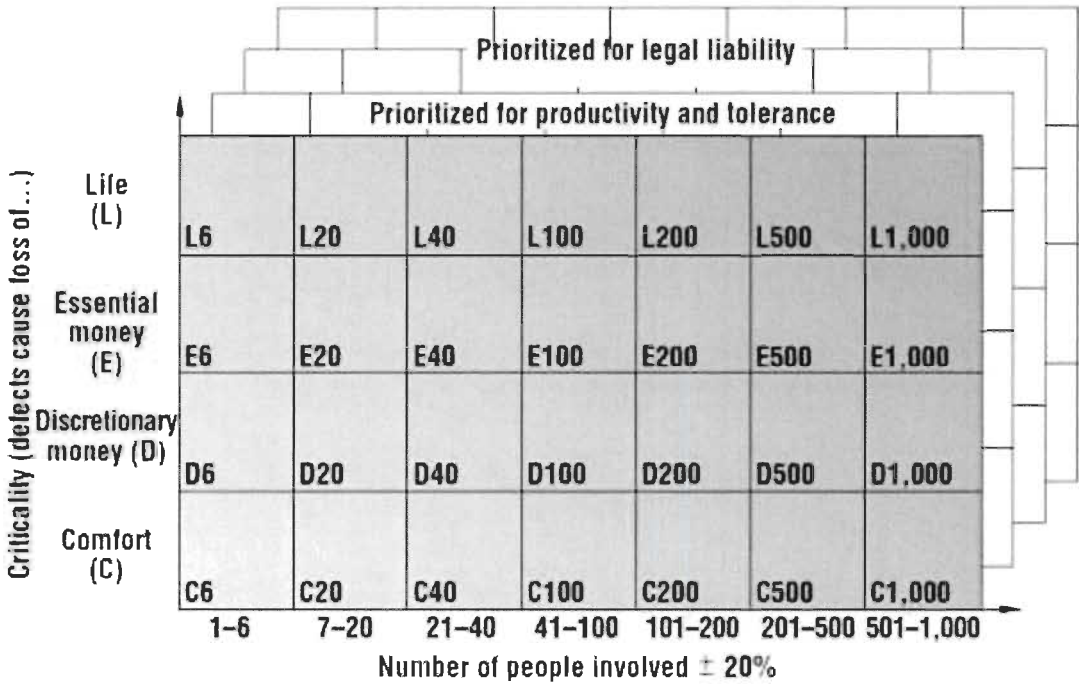

Figure 19 – Couverture Crystal de différents types de projets

(Crédits d'image: Cockburn, 2000)

En parlant de Crystal Clear, Cockburn sélectionne et définit cinq stratégies (Cockburn, 2005):

1. ''Exploratory 360°'', partie de la charte de projet;

2. ''Early Victory'', une stratégie de gestion de projet;

3. ''Walking Skeleton'', une stratégie conjointe d'architecture et de gestion de projet;

4. Réarchitecture incrémentale, une stratégie liée à ''Walking Skeleton''; l'équipe applique l'idée de développement incrémental à la révision de l'infrastructure ou de l'architecture ainsi que des fonctionnalités finales du système.

5. Les radiateurs d'information, une stratégie de communication.

Toujours en parlant de Crystal Clear, il définit neuf techniques:

1. ''Methodology Shaping'', collecte d'informations sur les expériences antérieures et utilisation pour élaborer les conventions de démarrage;

2. ''Reflection Workshop'', un format d'atelier particulier pour l'amélioration de la réflexion (Reflective Improvement);

3. Blitz Planning, qu'il appelle aussi parfois une ''jam session'' de planification de projet (comme dans le jazz) pour souligner sa nature collaborative, une technique rapide et collaborative de planification de projet;

4. Estimation Delphi, une façon de proposer une estimation initiale pour l'ensemble du projet;

5. Stand-ups quotidiens, un moyen rapide et efficace de transmettre quotidiennement des informations à l'équipe;

6. ''Agile Interaction Design'', une version rapide de la conception centrée sur l'utilisation;

7. Process Miniature, une technique d'apprentissage;

8. Programmation côte à côte (Side-by-Side Programming), une alternative moins intense à la programmation en binôme;

9. ''Burn Charts'', un moyen efficace de planifier et de rapporter les progrès, particulièrement adapté pour une utilisation sur les radiateurs d'information (Information Radiators).

Lorsqu'il parle des tableaux burn-down, Cockburn parle de l'importance de disposer du temps le plus court possible pour vérifier l'état du projet afin d'éviter ce qu'il appelle ''la période d'interdiction'' (blackout period) (Cockburn, 2005). Comme nous pouvons le voir sur la *Figure 20*, plus l'unité est petite, plus la période de black-out initiale est courte.

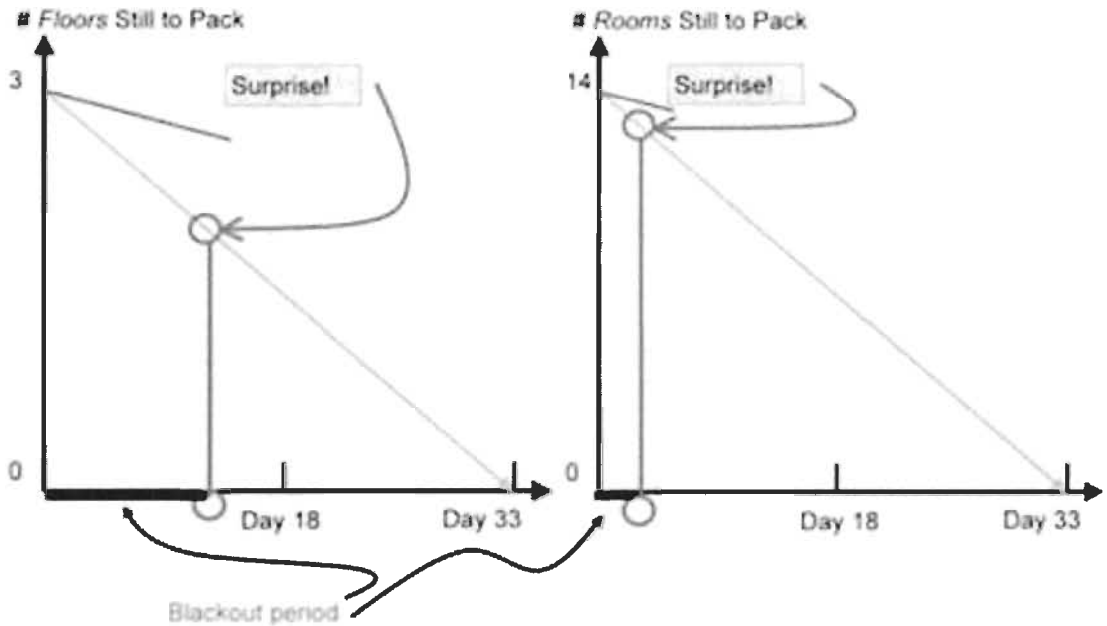

Figure 20 – Burn-down charts pour deux unités de base différentes

(Crédits d'image: Cockburn, 2005)

Des équipes interfonctionnelles sont utilisées dans Crystal pour réduire les livrables et améliorer la communication locale (Cockburn, 2002). Crystal Clear profite de la petite taille de l'équipe et de la proximité pour renforcer la communication étroite (Cockburn, 2005). La date de fin d'une itération est généralement considérée comme inamovible, une pratique appelée ''timeboxing''. Il est préférable que l'équipe remette tout ce qu'elle a terminé à la fin du délai, puis de reporter la date de fin.

Le meilleur des groupes Crystal Clear utilise des tests constants et des tests de régression entièrement automatisés (Cockburn, 2005). Les tests automatisés, la gestion de la configuration et l'intégration fréquente sont combinés en

intégration et test continus, de sorte que l'équipe détecte les erreurs de niveau d'intégration en quelques minutes. La livraison fréquente (incrémentielle) facilite un retour d'information rapide et riche sur le fait que le projet peut fonctionner avec très peu d'autres structures. La livraison d'un prototype ne doit pas être considérée comme une itération ou un cycle; chaque cycle de livraison doit produire du code testé et en cours d'exécution qui est, au moins en principe, d'une certaine utilité pour un utilisateur. Toutes les quelques semaines, une fois par mois ou deux fois par cycle de livraison, les gens se réunissent dans un atelier de réflexion ou une rétrospective d'itération pour discuter de la façon dont les choses fonctionnent. Cockburn recommande également l'utilisation d'un développement piloté par les tests. Les équipes peuvent essayer, sous différentes formes: programmation en binôme, tests unitaires et développement piloté par les tests.

La technique Scrum backlog est souvent utilisée sur Crystal Clear car une partie naturelle de Crystal Clear est intégrer certains éléments d'autres méthodologies (Cockburn, 2005).

Crystal s'appuie sur les meilleures pratiques qui existent depuis de nombreuses années, telles que le développement itératif, l'intégration continue, les user stories et bien d'autres (Kroll et al., 2003).

2.3.10. FDD (Feature-Driven Development)

Le développement basé sur les fonctionnalités (Feature-Driven Development - FDD) est une approche directe pour produire des systèmes qui utilisent des méthodes simples et faciles à comprendre et à mettre en œuvre, des techniques de résolution de problèmes et des directives de reporting fournissant à chaque partie prenante d'un projet les informations dont il a besoin pour créer des décisions judicieuses et opportunes (Palmer et al., 2002). Il a été développé pour répondre aux besoins spécifiques d'un grand projet de développement logiciel (Project Management Institute, 2017b).

FDD est un processus à courte itération basé sur le modèle. Cela commence par l'établissement d'une forme globale du modèle. Ensuite, il se poursuit avec une série d'itérations de ''conception par fonctionnalité, construction par fonctionnalité'' de deux semaines (Coad et al., 1999).

Le développement basé sur les fonctionnalités est un processus de développement logiciel adaptatif et Agile qui est (Palmer et al., 2002):

- Hautement itératif
- Souligne la qualité à chaque étape
- Fournit des résultats de travail fréquents et tangibles
- Fournit des informations de progression et d'état précises et significatives avec un minimum de surcharge et d'interruption pour les développeurs
- Est apprécié par les clients, les gestionnaires et les développeurs

Les clients l'apprécient car ils voient les résultats au début du projet et obtiennent des rapports d'étape clairs et faciles. Il est apprécié par les managers car ils obtiennent des informations complètes et exactes sur l'ensemble du projet pour le mener correctement. Il est apprécié par les développeurs car les petites équipes et les courtes itérations leur donnent des résultats fréquents de leur travail.

Le développement basé sur les fonctionnalités est basé sur des rôles principaux, de support et supplémentaires.

Rôles principaux du développement basé sur les fonctionnalités:

- Gestionnaire de projet
- Architecte en chef
- Directeur du développement
- Programmeur en chef
- Propriétaire de classe
- Expert de domaine

Rôles de support du développement basé sur les fonctionnalités:

- Gestionnaire de domaine
- Release Manager
- Avocat ou gourou des langues

- Ingénieur de construction
- Outilleur
- Administrateur du système

Rôles supplémentaires du développement basé sur les fonctionnalités:

- Testeurs
- Déployeurs
- Rédacteurs techniques

La méthode de développement basé sur les fonctionnalités recommande certaines bonnes pratiques d'ingénierie logicielle telles que:

- Modélisation d'objets de domaine
- Développement par fonctionnalité
- Propriété individuelle de classe
- Équipes de fonctionnalité
- Inspections
- Builds régulières
- Gestion de la configuration
- Rapport/visibilité des résultats

FDD utilise les niveaux de test suivants: test unitaire, test d'intégration et test système (Abrahamsson et al., 2003). L'utilisation de tests automatisés

minimise les tâches manuelles et répétitives afin que les testeurs puissent se concentrer sur leur travail de vérification de la fonctionnalité du système (Palmer et al., 2002). La saisie des cas de test dans un projet FDD provient principalement de la liste des fonctionnalités et des procédures pas à pas du domaine, mais peut également provenir de tout ce qui est utilisé comme exigences pendant le développement, comme les cas d'utilisation existants, les maquettes ou prototypes d'écran, les spécifications fonctionnelles, manuels d'utilisation, documents de politique d'affaires, etc.

Sur FDD, il est préférable d'utiliser une approche collaborative plus que l'ancienne approche de faire un travail individuel et de le jeter par-dessus le mur à la prochaine victime (Palmer et al., 2002). La plupart des développeurs peuvent gérer l'appartenance à deux ou même trois équipes de fonctionnalités simultanément pendant une courte période. L'intégration continue se produit également sur FDD, où des constructions de système continues ou régulières sont effectuées, exigeant également des tests continus ou réguliers. FDD est généralement associé à une programmation en binôme et à des tests unitaires extensifs.

La taille des fonctionnalités est généralement petite, suffisamment petite pour être mise en œuvre en deux semaines au maximum, ce qui signifie un développement incrémental et une livraison incrémentielle. Cela améliore la confiance des clients dans le projet et leur permet de fournir rapidement des

informations précieuses (Palmer et al., 2002). FDD fonctionne bien avec les jalons de timebox.

2.3.11. **Manifeste Agile** (Beck et al., 2001)

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.

2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.

3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.

4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.

6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.

7. Un logiciel opérationnel est la principale mesure d'avancement.

8. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.

9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.

10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.

11. Les meilleures architectures, spécifications et conceptions émergent d'équipes autoorganisées.

12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

2.3.12. Durabilité

La durabilité est le moyen de créer et de maintenir les conditions dans lesquelles l'homme et la nature peuvent exister en harmonie productive pour soutenir les générations présentes et futures (EPA, 2011). Nous comprenons que les humains ne sont pas censés détruire la capacité de la nature à se préserver en éteignant les ressources nécessaires pour cela.

La manière la plus simple de voir cela est l'utilisation par l'humanité de l'énergie provenant des ressources naturelles ne pourrait pas épuiser ces ressources, de sorte qu'elles devraient soit utiliser moins de ressources, soit trouver de nouvelles sources d'énergie renouvelables.

On peut définir le développement durable comme un moyen de développer un produit en utilisant moins d'énergie ou en utilisant le moins d'énergie possible. Le produit durable ne fait pas l'objet de ce travail. Le développement doit être durable pour accélérer le processus de développement et utiliser moins de ressources. En utilisant moins de ressources, le coût du développement est le minimum possible.

Nous pouvons également apprendre des processus durables comment mieux utiliser l'énergie pour augmenter les performances des projets aéronautiques.

Il est attendu d'un projet que ses gestionnaires travaillent pour maintenir le budget du projet. Concrètement, c'est une bonne idée d'essayer de minimiser les coûts de développement pour garder le budget.

Le guide PMBOK définit le processus de maîtriser les coûts comme le processus de suivi de l'état du projet pour mettre à jour les coûts du projet et gérer les modifications de la base de référence des coûts. Le principal avantage de ce processus est que la base de référence des coûts est maintenue tout au long du projet (Project Management Institute, 2017a).

Nous pouvons supposer que l'utilisation des pratiques de durabilité des Méthodes Agiles peut aider à minimiser les coûts du projet et à maintenir le budget.

2.3.13. Pratiques des méthodes Agiles

Le *Tableau 3* est proposé sur la base d'une recherche sur les pratiques ou techniques les plus utilisées dans chaque méthodologie Agile. Les pratiques sont brièvement développées dans la description de la méthodologie à travers les sections du présent travail.

Tableau 3 - Pratiques Agiles utilisées dans les Méthodes Agiles

Pratique	Iterative and incremental	Evolutive	ASD	RAD	RUP	DSDM	Scrum	XP	Crystal	Feature driven
Test automatisé	■			■	■	■	■	■	■	■
Test manuel	■	■	■	■	■	■	■	■	■	■
Backlog	■					■	■		■	
Burn-down chart							■		■	
Clean rooms				■						
Collaboration	■		■	■	■	■	■	■	■	■
Concurrence	■		■							■
Intégration continue	■		■		■		■	■	■	■
Équipe interfonctionnelle	■		■		■	■	■		■	
Réunion quotidienne					■	■	■	■	■	
Feedback	■	■	■	■	■	■	■	■	■	■
Livraison incrémentielle	■	■		■	■	■	■	■	■	■
Développement incrémental	■	■		■	■	■	■	■	■	■
Prototypage		■		■	■	■	■	■	■	■
Développement itératif	■		■	■	■	■	■	■	■	■
Kanban							■	■		
Programmation en binôme					■			■	■	■
Planning poker							■			
Outils de développement rapide				■		■				
Retrospective						■	■		■	
Développement de courte durée		■	■	■	■	■	■	■	■	■
Petites équipes	■		■	■	■	■	■	■	■	■
Sprint							■	■		
Planification de sprint							■	■		
Revue de sprint							■	■		

Développement piloté par les tests										
Timeboxing										
Test de l'unité										
User story										

2.4. Développement d'avions

Il est vrai que les premiers avions fabriqués dans l'histoire de l'humanité étaient faits à la main. Ils étaient tous des prototypes, plus construits comme preuve de concept et d'expériences scientifiques, ils n'avaient donc pas besoin de processus de production.

Les premières tentatives de pilotage d'un avion, un engin "plus lourd que l'air", n'ont été faites que longtemps après avoir été imaginées ou même conçues. Au 19e siècle, des scientifiques s'inquiétaient déjà de toute la science nécessaire à la production de ces engins, comme le mathématicien Isidore Didion en 1837 a déclaré que "la navigation aérienne n'aura aucune chance de succès si nous ne trouvons pas un moteur capable de produire une force motrice dont rapport au poids de la machine dont il aurait besoin pour être soutenu, est plus important que dans la vapeur, ou chez l'homme et la plupart des animaux. " (Didion 1837)

Il y avait Sir George Cayley qui a travaillé sur la physique du vol et développé en 1799 le concept de la machine volante à voilure fixe avec portance,

propulsion et contrôle (Crouch 2018), Cayley a conçu le premier planeur réussi à transporter un homme dans les aires (Crouch, 2019).

L'histoire de l'avion a suivi de près l'histoire des moteurs, et une machine volante de machine à vapeur, appelée "Aerial Steam Carriage", également "Ariel", a été proposée en 1843 (Wright, 1945).

Le moteur de l'avion Frères Wright 1903 avait déjà de l'essence comme carburant (Wright, 2018). Sans faire évoluer les moteurs en carburant liquide, il serait impossible de réduire le poids des moteurs et, d'avoir un poids total de l'avion inférieur à celui nécessaire pour permettre le vol de l'avion.

Les frères Wright ont dû développer leur propre moteur car il n'y avait pas de moteur sur le marché avec leurs besoins: 10 CV avec moins de 200 livres de poids. Il avait encore besoin d'un radiateur à eau pour garder le moteur au frais.

À cette époque, les exigences étaient simplement de permettre de mettre l'homme en l'air au moyen d'un véhicule de vol à moteur. Pour mettre plus d'un homme dans les airs (pilote et passagers), ils devaient augmenter la relation puissance/poids du moteur comme le moteur rotatif inventé par l'australien Lawrence Hargrave qui a propulsé les combattants les plus rapides de la Première Guerre mondiale avec 290 livres et 160 CV et pas radiateur à eau nécessaire - il utilise le flux d'air provenant de l'hélice pour refroidir le moteur.

Seuls les moteurs radiaux développés (perfectionnés) par Frederick Brent Rentschler ont rendu possible les avions de ligne, construits dans sa société Pratt & Whitney (avec 650 livres et 35 fois plus de puissance que celui de Wright

Brothers) comme le moteur Ford Tri 1920, l'un des tout premiers avions de ligne - il lui fallait une piste de deux mille et demi de pieds. L'hélice à pas variable a permis à la portée de l'avion d'être plus longue - 3000 miles. Le 1930 Flight Boat utilisait une hélice à pas variable - mais la piste nécessaire au décollage et à l'atterrissage était encore trop longue, donc le décollage et l'atterrissage ont été effectués au-dessus de l'eau.

Le développement du turboréacteur a permis à la piste d'être suffisamment petite pour permettre le décollage et l'atterrissage au sol, inventé en 1929 par le lieutenant d'aviation et pilote d'essai de la RAF Frank Whittle.

Le Gloster Meteor a été le premier chasseur à réaction opérationnel, mis en service en 1944.

Les turboréacteurs ont permis de meilleures performances de vol principalement pour trois raisons: la première étant le meilleur rapport puissance/poids. Le second, le moteur à hélice est inefficace à plus de 400 miles/heure, comme la version finale de la Lockheed Constellation qui volait à 320 miles/heure. Le Havilland Comet propulsé par des turboréacteurs a démarré à 520 miles/heure. Et le troisième, l'efficacité du turboréacteur est meilleure avec des altitudes plus élevées car moins de carburant est nécessaire pour voler.

Un autre élément important de l'avion, la structure, a également évolué, non seulement pour résoudre les problèmes d'aérodynamique, mais aussi pour le poids et la résistance. Les structures ont évolué de l'alliage d'aluminium dans les

années 1930 à un mélange d'alliage d'aluminium et de matériau composite dans les années 1990 (Paul 2004).

2.4.1. Industrie aéronautique actuelle

2.4.1.1. Cycle de vie du développement des aéronefs

Chaque compagnie aéronautique a ses propres méthodes et processus, mais tous ont une certaine variation des phases suivantes: faisabilité, concept, conception préliminaire, conception détaillée, essais et production (*Figure 21*).

Figure 21 – Cycle de vie de développement de l'avion

Phase de faisabilité:

La phase de faisabilité est une phase importante lorsque les objectifs du projet sont définis avec les options disponibles pour atteindre ces objectifs (Building, 2014). Il est défini que la phase de faisabilité commence par une compréhension conceptuelle des besoins du client et se développe jusqu'à la

définition des exigences du client. Au cours de cette phase, il est également identifié et évalué toutes les options disponibles et la conclusion est la décision de poursuivre ou non le projet. Ils définissent les résultats de la phase de faisabilité comme:

- définition des objectifs du client
- évaluation des différentes options
- une meilleure ligne de conduite recommandée pour atteindre les objectifs
- démonstration de la viabilité financière de la solution préférée
- confirmation d'un descriptif de projet
- business case (produite par le client)
- approbation par le client au niveau de la direction des propositions de projet (généralement basée sur une business case)
- décision du client de passer ou non à l'étape suivante
- préparer le périmètre des services du consultant principal

Dans le cas de l'industrie aéronautique, il n'y a pas de client réel à cette phase pour la plupart des projets, le client est donc représenté par une commission de marché compétente qui définit les besoins du marché au moment du lancement du projet. La quantité de passagers, la taille de la cargaison et l'autonomie de vol (distance maximale parcourue entre le décollage et l'atterrissage) sont parmi les choses les plus importantes considérées par eux. De nombreux éléments peuvent être explorés dans l'étude de marché tels que la

fin de la vie utile de certains avions avec une grande flotte d'opération, certaines technologies aéronautiques sont devenues obsolètes ou une nouvelle technologie propose de grandes économies dans l'opération.

Au cours de cette phase, le document d'étude de faisabilité est élaboré et doit comprendre (Building, 2014):

• le périmètre de l'enquête (d'après le résumé du projet), y compris l'établissement des objectifs de service et des objectifs financiers

• études sur les exigences et les risques

• consultation publique (étude marketing)

• consultation des parties prenantes et des tiers

• exigences ou contraintes légales / statutaires / de planification

• stratégie de gestion des risques

• estimations des capitaux et des coûts d'opération

• calendrier de développement principal

Pendant la phase de faisabilité, il est décidé si le projet consistera en des modifications mineures d'un avion existant, des modifications majeures ou une conception entièrement nouvelle.

Phase de concept:

Selon les pratiques du cycle de vie des avions d'Airbus (Airbus, 2017), la phase de concept est responsable de plusieurs décisions majeures, comme:

- Validation des objectifs de performance des avions
- Sélection de l'architecture des systèmes et de la structure
- Autorisation de ramp-up des ressources
- Confirmation du lancement industriel

Dans cette phase sont également définis des livrables clés tels que:

- La conception de l'avion
- Concepts de construction et de composants
- Marges de performance
- Marges commerciales

Les fournisseurs sont sélectionnés dans cette phase. Les documents des exigences des système sont produits au cours de cette phase.

Phase de conception préliminaire:

L'architecture du système est définie lors de la phase de conception préliminaire. Il est également important de décrire les fonctions du système (Blanchard et al., 2014), car la description fonctionnelle du système est développée pour servir de base à l'identification des ressources nécessaires au système pour accomplir sa mission. Selon eux, la conception préliminaire a pour but de démontrer que le concept de système sélectionné sera conforme aux spécifications de performance et de conception, et qu'il peut être produit et / ou

construit avec les méthodes disponibles, et que les contraintes de coût et de l'échéancier établies peuvent être respectées. En outre, la conception préliminaire peut avoir les produits suivants:

- analyse fonctionnelle et allocation des besoins au niveau du sous-système et au-dessous

- identification des critères de conception comme contribution au processus de conception

- application de modèles et de méthodes analytiques dans la conduite des compromis de conception

- réalisation de revues formelles de conception tout au long du processus de développement du système

- planification de la phase de conception détaillée.

La conception préliminaire traduit les besoins du projet dans les documents de spécification du système.

Les documents des exigences des systèmes sont examinés et approuvés au cours de cette phase.

Le PDR (Preliminary Design Review), Revue de Conception Préliminaire, clôt cette phase.

Phase de conception détaillée:

La phase de conception détaillée est chargée de décomposer l'architecture du système en composants. La spécification des composants et les dessins sont ensuite produits au cours de cette phase. Les documents de qualification des composants sont produits lors de cette phase.

Le CDR (Critical Design Review), Revue critique de la conception, clôt cette phase.

Phase de test (essais):

La phase de test comprend généralement la qualification, l'intégration des systèmes, les essais au sol, la campagne d'essais en vol et la certification des aéronefs.

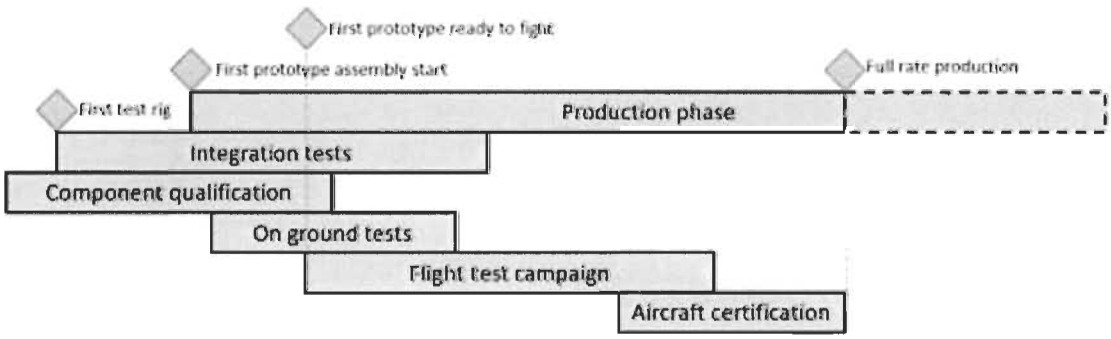

Figure 22 – Phase d'essai et de production de l'avion

Qualification:

La qualification des composants doit être obtenue avant la fin de la campagne d'essais en vol. Ce n'est qu'après que le composant est pleinement qualifié qu'il peut être testé en vol.

Les composants critiques ne sont pas censés être lancés en vol sans être pleinement qualifiés. Les tests de qualification sont des tests qui devraient être effectués sur les composants de l'aéronef concernant les conditions environnementales. Ces conditions couvrent toutes les situations possibles où l'avion devrait fonctionner, en vol ou au sol. Pour être testé, chaque composant d'aéronef doit être évalué pour une gamme de conditions qui peuvent se produire pour un composant donné situé à un endroit à l'intérieur ou à l'extérieur de l'aéronef. Ces conditions comprennent: la température, l'altitude, la pression, les vibrations, l'humidité, les vibrations, les chocs, les explosions, le sable, la poussière, les fluides, les fungus, les champs électromagnétiques, le sel, le brouillard, les pics de puissance et les interruptions, les interférences audio et radiofréquence, la foudre, le givrage, inflammabilité et autres. Pour être qualifié, il doit être prouvé que le composant donné doit fonctionner sans défaillance ni dégradation des performances dans le cas des gammes de conditions qui devraient se produire dans la vie normale du composant lors de son utilisation à bord de l'aéronef. Les niveaux de performances dégradées ou d'échec doivent être définis. Par exemple, un composant donné ne doit pas s'arrêter de fonctionner lorsque la température est de -10 degrés à 40 degrés Celsius. Un

autre composant doit enregistrer l'état actuel en cas de panne d'alimentation à zéro dans un intervalle de temps de cent millisecondes.

Intégration de systèmes:

L'intégration des systèmes commence bien avant la construction du prototype sur les bancs d'essai des systèmes (systems test rigs). Avant que les composants soient assemblés dans l'avion, tous les composants subissent un test d'intégration sur des plates-formes qui représentent des parties de l'avion. Dans ces plates-formes, les composants du même système d'avion sont testés pour les signaux électriques, accouplement et stress mécaniques. Souvent, des composants de différents systèmes d'aéronefs sont ajoutés à la plate-forme pour tester afin de vérifier s'ils sont compatibles, de manière mécanique et électrique.

Une grande majorité des problèmes d'intégration devraient être constatés avant les premiers vols d'avion. Si un problème d'intégration n'est pas résolu avant le premier vol du prototype, il peut être supprimé en évitant la fonctionnalité d'être utilisé pendant le vol, au cas où il ne serait pas obligatoire pour le vol, en d'autres termes, le décollage, le vol et l'atterrissage en toute sécurité sont assuré. Il est très courant d'avoir les premiers vols avec de nombreuses dérogations, et l'équipage de conduite doit réviser la liste des dérogations avant les vols.

Essais au sol:

De nombreux tests de systèmes d'aéronefs peuvent être effectués sans vol et peuvent être effectués sur les plates-formes ou dans le prototype partiellement

assemblé. Par exemple, de nombreux tests de systèmes de carburant peuvent être effectués dans l'avion au sol en ayant simplement les réservoirs d'aile entièrement montés. L'avion peut ainsi être placé dans les nombreuses attitudes nécessaires pour tester le comportement des réservoirs de carburant en simulant les attitudes attendues de l'avion en vol.

Campagne d'essais en vol:

La campagne d'essais en vol commence lorsque le premier prototype est prêt à voler. Pour cela, des conditions minimales du système doivent être atteintes pour assurer un décollage, un vol et un atterrissage en toute sécurité. Il est typique pour le premier vol que certains systèmes non critiques soient désactivés. Il est évident que tous les problèmes doivent être résolus avant la campagne d'essais en vol afin que l'avion puisse être entièrement testé en vol. La campagne d'essais en vol doit se terminer avant le certificat de type. La plupart des tests en vol sont utilisés pour prouver aux autorités que la conception de l'aéronef est conforme aux exigences de navigabilité.

Certification des aéronefs:

Le certificat de type est un document délivré à la fin de la certification de l'aéronef par une autorité réglementaire qui signifie qu'un aéronef donné est apte à un vol en toute sécurité (navigabilité). Un avion donné n'est pas autorisé à voler à l'intérieur d'un certain pays ou d'une certaine région sans ce document. Le

processus d'obtention de cette autorisation dépend de l'agence de navigabilité, mais consiste généralement à démontrer la conformité de cette agence aux normes obligatoires par des tests ou des analyses. Tous les documents produits sont envoyés aux autorités pour examen et les autorités visitent les avions avant le certificat de type. Il fait également partie de ce processus la communication entre les autorités et le constructeur aéronautique.

Phase de production:

La phase de production commence par l'assemblage du premier prototype. En règle générale, 1 à 3 prototypes sont construits pour être utilisés dans la campagne d'essais en vol et pour la certification. Certaines sociétés traitent les prototypes comme les premiers avions de série à être livrés aux clients, car la société n'a aucun intérêt à conserver les prototypes, surtout lorsque le prix de l'avion est élevé. La phase de production a la phase d'intégration qui est la première fois que tous les composants sont assemblés pour fonctionner ensemble - les tests d'intégration ont alors lieu. Une fois les tests d'intégration au sol réussis, la campagne d'essais en vol peut commencer. La qualification est une étape importante qui doit être finalisée avant la livraison des avions.

Production préliminaire:

La phase de production préliminaire commence lorsque le premier prototype commence à être assemblé et se termine avec le certificat de type.

Cette phase est importante du point de vue de la construction du calendrier de production. Bien que de nombreux problèmes se produisent retardant la fin de l'assemblage du prototype, nous pouvons avoir une bonne idée de ce qui devrait être un calendrier approprié, en connaissant mieux le temps d'assemblage de certains systèmes dans l'avion.

2.4.1.2. Agences Gouvernementales de Navigabilité et Documentation

Le développement d'un aéronef est devenu une activité très complexe et il existe un certain nombre d'agences qui produisent de la documentation afin de fournir des conseils à l'industrie aéronautique.

Chaque pays ou région a sa propre agence de navigabilité, comme la FAA (Federal Aviation Administration), SAE International (Society of Automotive Engineers) et RTCA (Radio Technical Cornmission for Aeronautics) aux États-Unis d'Amérique, JAA (Joint Aviation Authorities) et EASA (Agence Européenne de la Sécurité Aérienne) en Europe, ANAC (Agência Nacional de Aviação Civil) au Brésil, OACI (Organisation de l'Aviation Civile Internationale) pour les Nations Unies, TCAC (Aviation Civile de Transports Canada) au Canada et autres.

Moir (Moir et al., 2008) a défini une liste non exhaustive d'agences avec une documentation importante largement appliquée dans l'industrie aéronautique. La liste est:

• Society of Automobile Engineers (SAE):

- ARP 4754 Lignes directrices pour le développement des aéronefs et systèmes civils.

- ARP 4761 Lignes directrices et méthodes pour la conduite du processus d'évaluation de la sécurité des systèmes et équipements aéroportés civils.

• Federal Aviation Authority (FAA - Autorité nord-américaine):

- 14 CFR 25.1309-1A Normes de navigabilité: Avions de la catégorie Transport - Équipement, systèmes et installation

- AC 25.1309 Circulaire d'information - Conception et analyse de systèmes

• Joint Airworthiness Authority (JAA - Autorité européenne):

- AMJ 25.1309 Joint Material Advisory - Conception et analyse de systèmes

• Air Transport Association (ATA):

- ATA-100

• Radio Technical Committee Association (RTCA):

- DO-178 Considérations logicielles dans la certification des systèmes et équipements aéroportés

- DO-254 Conseils en matière d'assurance de la conception du hardware électronique aéroporté

- DO-297 Integrated Modular Avionics (IMA) Conseils de développement et considérations de certification

- DO-160 Conditions environnementales et procédures d'essai pour l'équipement aéroporté

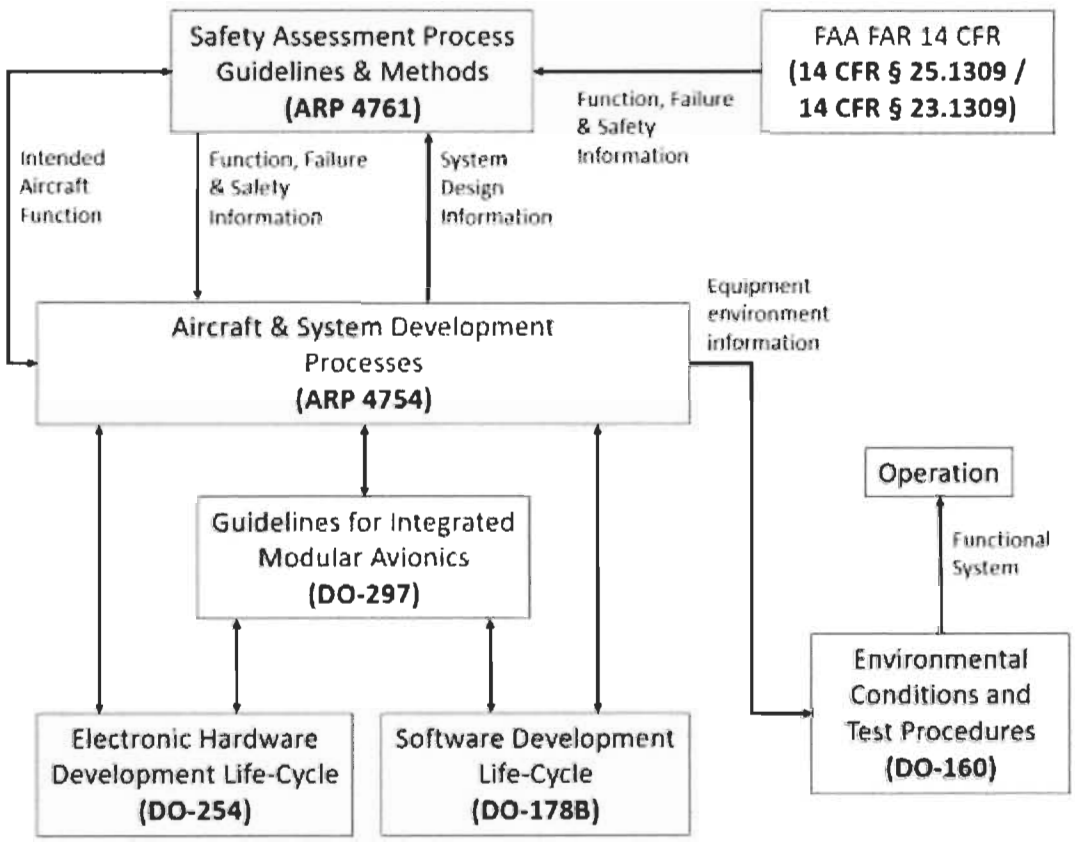

Figure 23 – Processus de développement du système selon ARP 4754

(Crédits d'image: ARP4754, 2010)

2.4.1.3. SAE ARP 4754

Bien que SAE signifie Society of Automotive Engineers, la société est connue sous le nom de SAE International, elle a des divisions pour de nombreuses industries et met l'accent sur les industries du transport comme l'aérospatiale, l'automobile et les véhicules utilitaires. L'ARP 4754 est signé par "SAE Aerospace, an SAE International Group", et compte des membres de comité contributeurs d'entreprises du monde entier comme Boeing, Embraer, Honeywell, Airbus, Rolls Royce, Dassault Aviation, Bell Helicopters, GE Aviation et bien d'autres.

Le SAE ARP 4754 discute du développement des systèmes de l'avion en incluant la validation des exigences et la vérification de la mise en œuvre de la conception pour la certification et l'assurance produit (ARP 4754, 2010). Il fournit également des directives mises à jour et élargies pour les processus utilisés pour développer des avions et des systèmes civils.

Structure de l'ARP 4754 (ARP 4754, 2010)

Chapitre 3: Planification du développement.

Chapitre 4: Processus de développement des aéronefs et des systèmes (développement / mise en œuvre de systèmes d'aéronef, développement / mise en œuvre de fonctions d'aéronef, assurance de développement d'aéronefs, architecture de système, conception / construction / intégration de hardware et de logiciels).

Chapitre 5: Processus intégrés (évaluation de la sécurité, attribution du niveau d'assurance de développement, capture / validation des exigences, vérification de la mise en œuvre, gestion de la configuration, assurance des processus, certification et coordination des autorités réglementaires).

Chapitre 6: Modification des aéronefs ou systèmes.

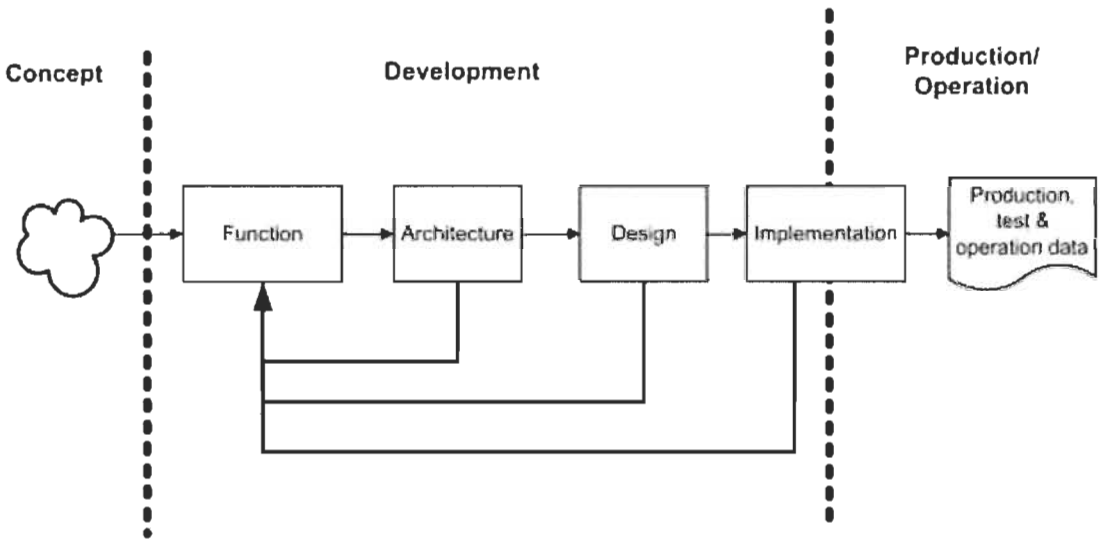

Figure 24 – Cycle de vie de développement selon ARP4754A

(Crédits d'image: ARP4754, 2010)

Niveau d'assurance de développement d'article (IDAL - Item Development Assurance Level)

L'Item Development Assurance Level (IDAL) est le niveau de rigueur des tâches d'assurance de développement à respecter lors du développement d'un article (ARP 4754, 2010).

Le niveau d'assurance de développement (DAL – Development Assurance Level) affecté à un élément est défini comme suit (ARP 4754, 2010):

- Niveau A: si une condition de défaillance catastrophique peut résulter d'une erreur de développement possible dans une fonction ou un élément de l'avion / système.

- Niveau B: si une condition de défaillance dangereuse / grave peut résulter d'une erreur de développement possible dans une fonction ou un élément de l'aéronef / système.

- Niveau C: si une condition de défaillance majeure peut résulter d'une erreur de développement possible dans une fonction ou un élément de l'avion / système.

- Niveau D: si une condition de défaillance mineure peut résulter d'une erreur de développement possible dans une fonction ou un élément de l'avion / système.

- Niveau E: s'il n'y a aucun effet sur la sécurité résultant d'une éventuelle erreur de développement dans une fonction ou un élément de l'aéronef / système.

ARP-4754 Cycle de vie du développement

L'ARP4754 définit les phases du cycle de vie du développement comme fonction, architecture, conception et implémentation (*Figure 24*).

Comme nous pouvons le voir dans la *Figure 24*, un certain degré d'itération est attendu en raison du retour des flèches aux phases précédentes.

2.4.1.4. SAE ARP 4761

L'ARP 4761 est signé par "SAE International, The Engineering Society for Advanced Mobility Land, Sea, Air and Space", et compte des membres de comités d'entreprises de partout dans le monde comme Boeing, Honeywell, Rolls Royce, Rockwell Collins Avionics, British Aerospace et beaucoup d'autres.

Le SAE ARP 4761 décrit les lignes directrices et les méthodes de réalisation de l'évaluation de la sécurité pour la certification des aéronefs civils, y compris les systèmes et équipements aéroportés. Ce document définit le processus d'évaluation de la sécurité et les méthodes d'analyse.

L'ARP 4761 définit également le cycle de vie du développement de produits comme les exigences (développement de concept), la conception (conception préliminaire et conception détaillée) et les tests (validation et vérification de la conception).

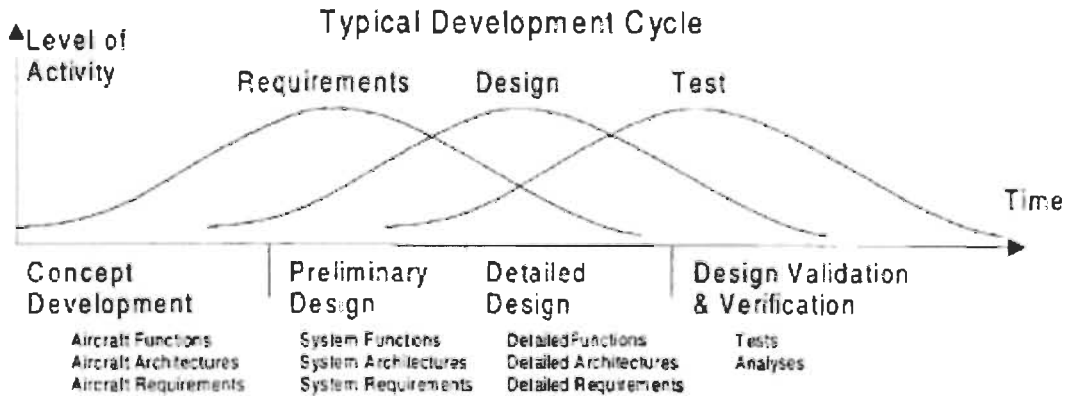

Figure 25 – Cycle de développement selon ARP 4761

(Crédits d'image: ARP4761, 1996)

Le SAE ARP 4761 décrit brièvement les évaluations, l'analyse et les diagrammes à travers ses chapitres et donne des instructions détaillées sur la façon de les documenter dans les annexes.

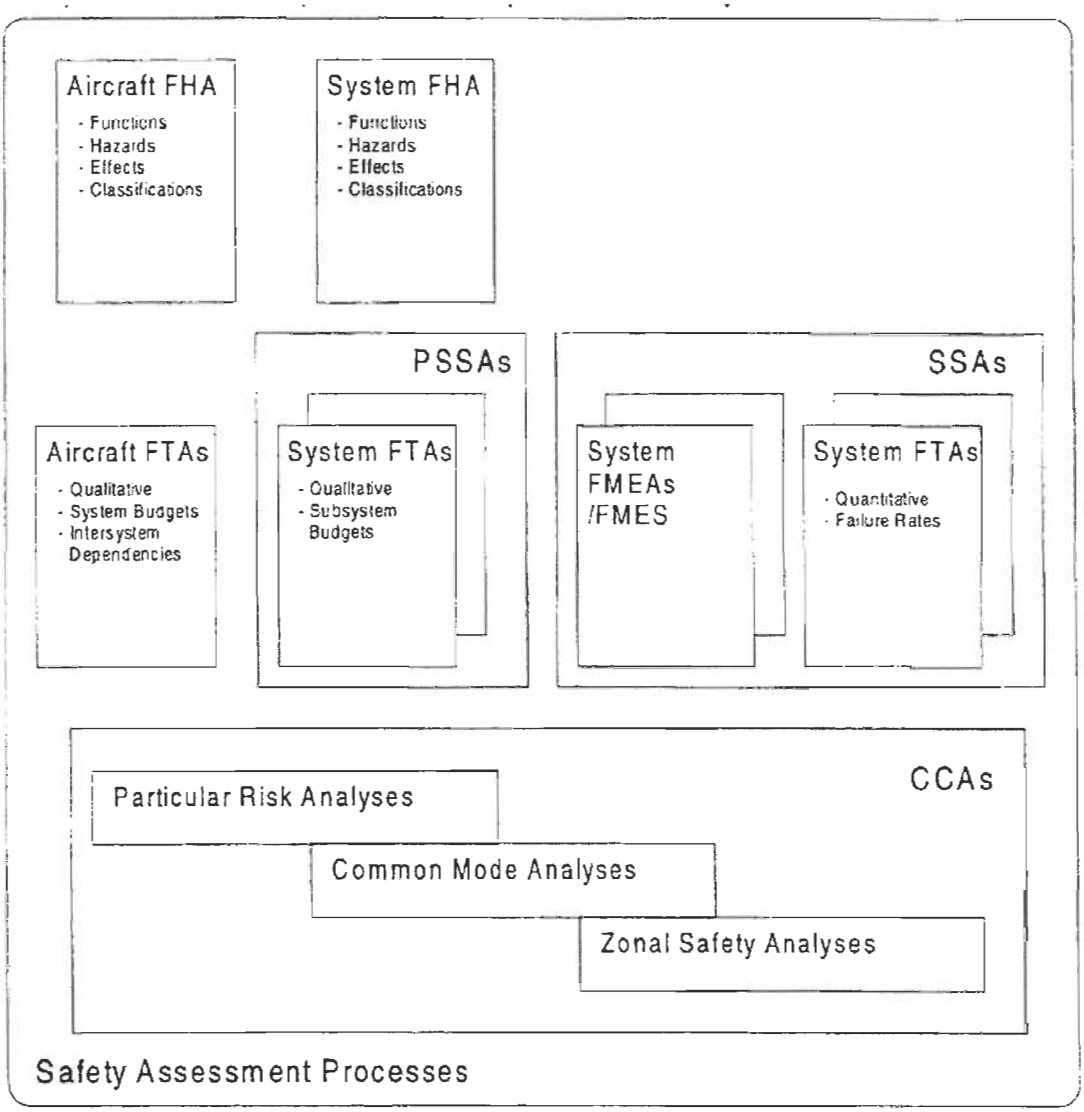

Figure 26 – Processus System Assessment selon ARP 4761

(Crédits d'image: ARP4761, 1996)

Processus d'évaluation de la sécurité:

- Évaluation des risques fonctionnels (FHA - Functional Hazard Assessment)

- Évaluation préliminaire de la sécurité du système (PSSA - Preliminary System Safety Assessment)

- Évaluation de la sécurité du système (SSA - System Safety Assessment)

Méthodes d'analyse de l'évaluation de la sécurité:

- Analyse de l'arbre de défaillance (FTA - Fault Tree Analysis)

- Diagramme de dépendance (DD - Dependence Diagram)

- Analyse de Markov (MA - Markov Analysis)

- Analyse des modes de défaillance et des effets (FMEA - Failure Modes and Effects Analysis)

- Résumé des modes de défaillance et des effets (FMES - Failure Modes and Effects Summary - Common Cause Analysis)

- Analyse des causes communes (CMA - Common Cause Analysis)

- Analyse de sécurité zonale (ZSA - Zonal Safety Analysis)

- Analyse des risques particuliers (PRA - Particular Risks Analysis)

- Analyse en mode commun (CMA - Common Mode Analysis)

2.4.1.5. 14 CFR 25.1309-1A

Ce document fait partie d'un ensemble de documents émis par la Federal Aviation Authority (FAA) appelé CFR (Code of Federal Regulations) et est attribué comme: "Title 14. Aeronautics and Space > Chapter I. FEDERAL AVIATION ADMINISTRATION, DEPARTMENT OF TRANSPORTATION > Subchapter C. AIRCRAFT > Part 25. AIRWORTHINESS STANDARDS:

TRANSPORT CATEGORY AIRPLANES > Subpart F. Equipment > subgroup General > Section 25.1309. Equipment, systems, and installations''.

Bien que court, ce document est central, très important et toujours référencé dans un autre document de référence comme une norme de navigabilité. La circulaire consultative AC 25.1309 est un document plus détaillé et elle est rédigée pour décrire divers moyens acceptables pour démontrer la conformité aux exigences du § 25.1309 du Federal Aviation Regulations (FAR). Alors que AC 25.1309 compte environ 6 700 mots, 14 CFR 25.1309-1A ne compte que 370 mots environ.

Le document 14 CFR 25.1309-1A est rédigé pour la catégorie 25 du transport aérien. Il existe une version similaire de ce document écrite pour la catégorie des avions de la partie 23, le 14 CFR 23.1309.

La catégorie des aéronefs de la partie 23 comprend les types normaux, utilitaires, acrobatiques et de navette (indiqués au 14 CFR 23.3) (14 CFR Part 23, 2020).

- Catégorie normale: neuf sièges passagers ou moins, masse maximale certifiée au décollage de 12 500 livres ou moins, et destinée à une utilisation non acrobatique.

- Catégorie utilitaire: neuf sièges passagers ou moins, masse maximale certifiée au décollage de 12 500 livres ou moins, et destinée à une utilisation acrobatique limitée.

- Catégorie acrobatique: neuf sièges passagers ou moins, masse maximale certifiée au décollage de 12 500 livres ou moins, et destinée à être utilisée sans restriction, autres que celles jugées nécessaires à la suite des tests en vol requis.

- Catégorie navette: est limitée aux avions multimoteurs à hélices qui ont une configuration de sièges, à l'exclusion des sièges de pilote, de 19 ou moins, et une masse maximale certifiée au décollage de 19 000 livres ou moins. L'opération de catégorie navette est limitée à toute manœuvre incidente au vol normal, aux décrochages (à l'exception des décrochages à fouet) et aux virages serrés, dans lesquels l'angle d'inclinaison ne dépasse pas 60 degrés.

La catégorie des avions de la partie 25 est prise en compte (Parson, 2015):

• Jets avec 10 sièges ou plus ou une masse maximale au décollage (MTOW) supérieure à 12 500 lb,

• Avions à hélices de plus de 19 sièges ou MTOW supérieur à 19 000 lb,

• Au moins deux moteurs,

• Piloté par au moins deux pilotes,

• ''Fail-safe'' - tout élément peut tomber en panne, mais le risque d'une telle défaillance provoquant un accident doit être extrêmement faible,

• Limite de charge de -1 à +2,5 Gs (ou jusqu'à +3,8 Gs, selon la masse au décollage de conçue).

14 CFR 25.1309-1A texte (14 CFR Part 25, 2020)

"§ 25.1309 Équipements, systèmes et installations.

(a) L'équipement, les systèmes et les installations dont le fonctionnement est requis par ce sous-chapitre doivent être conçus pour garantir qu'ils remplissent les fonctions prévues dans toutes les conditions de fonctionnement prévisibles.

(b) Les systèmes de l'avion et les composants associés, considérés séparément et en relation avec d'autres systèmes, doivent être conçus de telle sorte que -

(1) La survenance de toute condition de défaillance qui empêcherait le vol et l'atterrissage en toute sécurité de l'avion est extrêmement improbable, et

(2) La survenance de toute autre condition de défaillance qui réduirait la capacité de l'avion ou la capacité de l'équipage à faire face à des conditions d'opération défavorables est improbable.

(c) Des informations d'avertissement doivent être fournies pour alerter l'équipage des conditions de fonctionnement dangereuses du système et pour lui permettre de prendre les mesures correctives appropriées. Les systèmes, les commandes et les moyens de surveillance et d'avertissement associés doivent être conçus pour minimiser les erreurs de l'équipage qui pourraient créer des risques supplémentaires.

(d) La conformité aux exigences du paragraphe (b) de cette section doit être démontrée par l'analyse et, si nécessaire, par des essais au sol, en vol ou sur simulateur appropriés. L'analyse doit considérer –

(1) Modes de défaillance possibles, y compris les dysfonctionnements et les dommages provenant de sources externes.

(2) La probabilité de défaillances multiples et de défaillances non détectées

(3) les effets qui en résultent sur l'avion et ses occupants, compte tenu du stade de vol et des conditions d'opération, et

(4) Les signaux d'avertissement de l'équipage, les mesures correctives requises et la capacité de détecter les défauts.

(e) Pour démontrer la conformité aux paragraphes (a) et (b) de cette section en ce qui concerne la conception et l'installation du système électrique et de l'équipement, les conditions environnementales critiques doivent être prises en considération. Pour les équipements de production, de distribution et d'utilisation électriques requis par ou utilisés pour se conformer à ce chapitre, à l'exception des équipements couverts par des ordres techniques standard (Technical Standard Orders – TSO) contenant des procédures de test environnemental, la capacité à fournir un service continu et sécurisé edans des conditions environnementales prévisibles peut être démontrée par des tests environnementaux, analyse de conception ou référence à une expérience de service comparable antérieure sur d'autres aéronefs.

(f) L'EWIS doit être évalué conformément aux exigences du § 25.1709.

Comme nous pouvons le voir dans le texte, le CFR 25.1309 stipule que les équipements, systèmes et installations aéroportés doivent être conçus pour assurer les fonctions souhaitées dans toutes les conditions attendues. Les systèmes et composants de l'aéronef doivent être conçus de manière à ce que toute défaillance qui empêche un vol et un atterrissage en toute sécurité soit extrêmement improbable et que toute défaillance qui réduit la capacité de l'avion ou la capacité de l'équipage soit improbable.

Les définitions de ce qui est improbable ou extrêmement improbable peuvent être trouvées dans l'AC 25.1309 comme suit:

(1) Les conditions de défaillance probables sont celles ayant une probabilité supérieure à celle de l'ordre de 1 X 10-5.

(2) Les conditions de défaillance improbables sont celles ayant une probabilité de l'ordre de 1 X 10-5 ou moins, mais supérieure à celle de l'ordre de 1 X 10-9.

(3) Les conditions de défaillance extrêmement improbables sont celles ayant une probabilité de l'ordre de 1 X 10-9 ou moins.

Le CFR 25.1309 indique également la nécessité d'avertir l'équipage lorsque des conditions dangereuses se produisent, de sorte que les systèmes critiques doivent être surveillés et capables de donner des alarmes d'avertissement.

La conformité avec le paragraphe (b) du CFR 25.1309 doit être démontrée par analyse ou par des essais au sol, en vol ou sur simulateur.

Le paragraphe (e) indique la qualification des systèmes et des équipements, c'est-à-dire qu'ils doivent être testés en tenant compte des conditions environnementales de fonctionnement. "

14 CFR 23.1309-1A texte (14 CFR Part 23, 2020)

"Sec. 23.1309 - Équipements, systèmes et installations.

(a) Chaque équipement, chaque système et chaque installation:

(1) Lors de l'exécution de sa fonction prévue, ne doit pas nuire à la réponse, au fonctionnement ou à la précision de tout -

(i) L'équipement essentiel à un fonctionnement sécurisé; ou

(ii) Autre équipement, sauf s'il existe un moyen d'informer le pilote de l'effet.

(2) Dans un avion monomoteur, doit être conçu pour minimiser les risques pour l'avion en cas de dysfonctionnement ou de panne probable.

(3) Dans un avion multimoteur, doit être conçu de manière à prévenir les risques pour l'avion en cas de dysfonctionnement ou de panne probable.

(4) Dans un avion de la catégorie navette, doit être conçu pour se prémunir contre les dangers pour l'avion en cas de dysfonctionnement ou de panne.

(b) La conception de chaque équipement, de chaque système et de chaque installation doit être examinée séparément et en relation avec d'autres systèmes et installations d'avion pour déterminer si l'avion dépend de sa fonction pour assurer un vol et un atterrissage en toute sécurité et, pour les avions ne se limite pas aux conditions VFR (Règles de vol à vue), si la défaillance d'un système réduit considérablement la capacité de l'avion ou la capacité de l'équipage à faire face à des conditions d'opération défavorables. Chaque équipement, chaque système et chaque installation identifiés par cet examen comme étant ceux dont l'avion dépend pour un bon fonctionnement afin d'assurer un vol et un atterrissage en toute sécurité, ou dont la panne réduirait considérablement la capacité de l'avion ou la capacité de l'équipage pour faire face à des conditions d'opération défavorables, doit être conçu pour satisfaire aux exigences supplémentaires suivantes:

(1) Il doit remplir sa fonction prévue dans toutes les conditions de fonctionnement prévisibles.

(2) Lorsque les systèmes et les composants associés sont considérés séparément et par rapport à d'autres systèmes—

(i) La survenance de toute condition de défaillance qui empêcherait le vol et l'atterrissage en toute sécurité de l'avion doit être extrêmement improbable; et

(ii) La survenance de toute autre condition de défaillance qui réduirait considérablement la capacité de l'avion ou la capacité de l'équipage à faire face à des conditions d'opération défavorables doit être improbable.

(3) Des informations d'avertissement doivent être fournies pour alerter l'équipage des conditions de fonctionnement dangereuses du système et pour lui permettre de prendre les mesures correctives appropriées. Les systèmes, les commandes et les moyens de surveillance et d'avertissement associés doivent être conçus pour minimiser les erreurs de l'équipage qui pourraient créer des dangers supplémentaires.

(4) La conformité aux exigences du paragraphe (b) (2) de la présente section peut être démontrée par une analyse et, si nécessaire, par des essais au sol, en vol ou sur simulateur appropriés. L'analyse doit considérer—

i) Modes de défaillance possibles, y compris dysfonctionnements et dommages causés par des sources externes;

(ii) la probabilité de défaillances multiples et la probabilité de défaillances non détectées;

(iii) Les effets qui en résultent sur l'avion et les occupants, compte tenu de l'étape du vol et des conditions d'opération; et

(iv) Les signaux d'avertissement de l'équipage, les mesures correctives requises et la capacité de l'équipage à déterminer les défauts.

(c) Chaque équipement, chaque système et chaque installation dont le fonctionnement est requis par le présent chapitre et qui nécessite une alimentation électrique est une ‘’charge essentielle’’ sur l'alimentation électrique. Les sources d'alimentation et le système doivent pouvoir fournir les charges électriques suivantes dans des combinaisons de fonctionnement probables et pour des durées probables:

(1) Charges connectées au système de distribution électrique avec le système fonctionnant normalement.

(2) Charges essentielles après défaillance de—

(i) Tout moteur sur un avion bimoteur; ou

(ii) Deux moteurs sur un avion avec trois moteurs ou plus; ou

(iii) Tout convertisseur de puissance ou dispositif de stockage d'énergie.

(3) Charges essentielles pour lesquelles une autre source d'énergie est requise, selon le cas, par les règles de fonctionnement de ce chapitre, après toute défaillance ou mauvais fonctionnement d'un système d'alimentation, d'un système de distribution ou d'un autre système d'utilisation.

(d) Pour déterminer la conformité avec le paragraphe (c) (2) de cette section, les charges de puissance peuvent être supposées réduites selon une procédure de surveillance compatible avec la sécurité dans les types d'opérations autorisées. Il n'est pas nécessaire de prendre en compte les charges non

requises en vol contrôlé pour les deux moteurs en panne sur les avions à trois moteurs ou plus.

e) Pour démontrer la conformité à la présente section en ce qui concerne le système d'alimentation électrique et la conception et l'installation des équipements, les conditions environnementales et atmosphériques critiques, y compris l'énergie radiofréquence et les effets (directs et indirects) des coups de foudre, doivent être pris en considération. Pour les équipements de génération, de distribution et d'utilisation électriques requis ou utilisés pour se conformer à ce chapitre, la capacité à fournir un service continu et sécurisé dans des conditions environnementales prévisibles peut être démontrée par des tests environnementaux, une analyse de conception ou une référence à une expérience de service comparable antérieure sur d'autres avions.

(f) Tel qu'utilisé dans cette section, ''système'' fait référence à tous les systèmes pneumatiques, systèmes de fluides, systèmes électriques, systèmes mécaniques et systèmes de propulsion inclus dans la conception de l'avion, à l'exception des éléments suivants:

(1) Systèmes de propulsion fournis dans le cadre du moteur certifié.

(2) La structure de vol (une telle aile, l'empennage, les gouvernes et leurs systèmes, le fuselage, le support moteur et le train d'atterrissage et leurs accessoires principaux associés) dont les exigences sont spécifiques dans les sous-parties C et D de cette partie. "

Le texte du CFR 23.1309 est très similaire au CFR 25.1309, car il indique:

- Les systèmes et équipements aéroportés doivent fonctionner comme prévu et le résultat d'une défaillance donnée en fonction de sa probabilité est défini;

- Les moyens d'alerte et de surveillance;

- Conformité démontrée par analyse ou par des essais au sol, en vol ou sur simulateur, et

- Qualification des systèmes et équipements en fonction des conditions environnementales de fonctionnement.

En prenant l'exemple du document CFR 25 dans son ensemble, la partie principale du document est divisée en sous-parties suivantes:

- Sous-partie A - Généralités (§§ 25.1 - 25.5)

- Sous-partie B - Vol (§§ 25.21 - 25.255)

- Sous-partie C - Structure (§§ 25.301 - 25.581)

- Sous-partie D - Conception et construction (§§ 25.601 - 25.899)

- Sous-partie E - Groupe motopropulseur (§§ 25.901 - 25.1207)

- Sous-partie F - Équipement (§§ 25.1301 - 25.1461)

- Sous-partie G - Limites d'opération et informations (§§ 25.1501 - 25.1587)

- Sous-partie H - Systèmes d'interconnexion de câblage électrique (EWIS) (§§ 25.1701 - 25.1733)

- Sous-partie I - Règlement fédéral sur l'aviation spéciale (§ 25.1801)

Bien que le FAA 14 CFR 25.1309 soit un document très important à considérer dans le développement de l'avion (ou la version partie 23 pour les avions de type partie 23), non seulement cette section doit être considérée pour le développement, mais toutes les sections. Les CFR parties 23 et 25 sont considérées comme des normes de navigabilité et doivent être prouvées dans la conception de l'avion pour recevoir la certification par l'autorité de certification, la FAA, dans le cas où l'avion est certifié aux États-Unis.

2.4.1.6. AC 25.1309

La AC (circulaire consultative) 25.1309 de la FAA est un document qui décrit divers moyens acceptables de démontrer la conformité aux exigences des paragraphes 25.1309 (b), (c) et (d) du Federal Aviation Regulations (paragraphe 1, "Purpose").

La structure de la révision 1A de l'AC 25.1309 est la suivante:

1. Objet
2. Annulation (de la version précédente 1)
3. Applicabilité
4. Contexte
5. La conception "Fail-Safe" de la FAA
6. Définitions
7. Discussion

8. Techniques acceptables

9. Évaluation qualitative

10. Évaluation quantitative

11. Considérations opérationnelles et d'entretien

12. Guide étape par étape

L'AC 25.1309 clarifie les nombreux termes du 14 CFR 25.1309 et explique comment assurer sa conformité dans les projets d'avions. Il explique ce qu'est une condition de défaillance et comment elle est classée en mineur, majeur et catastrophique. Il définit la relation entre la probabilité de défaillance et sa gravité.

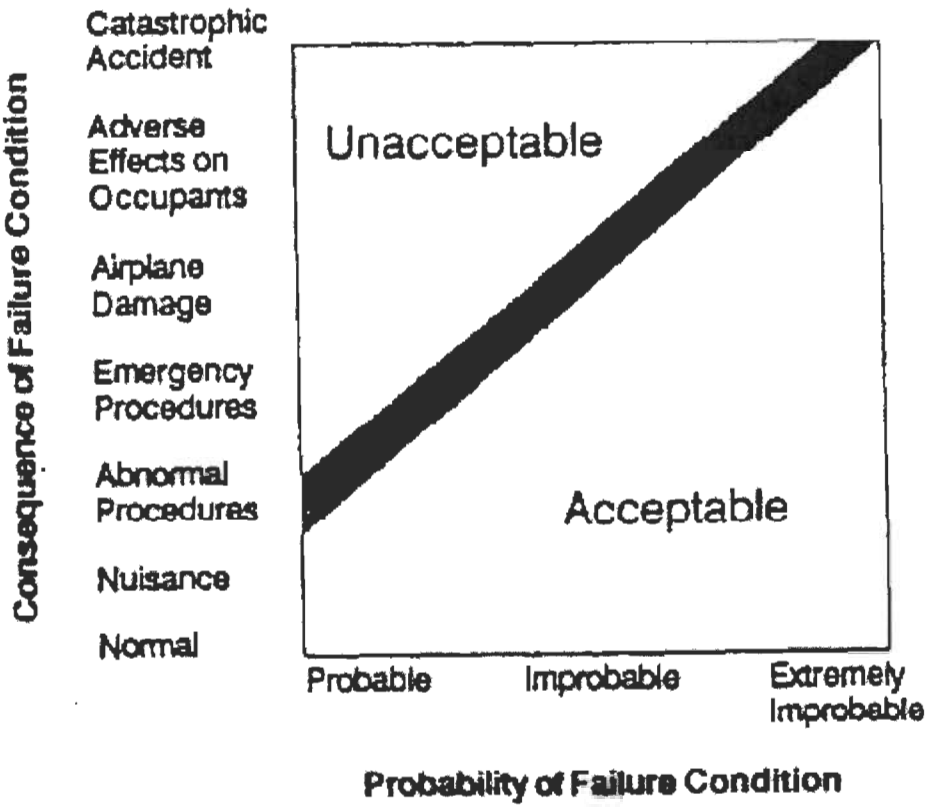

Figure 27 – Graphique de probabilité vs conséquence

(Crédits d'image: Federal Aviation Administration, 1988)

En conséquence, nous avons ce qui peut être considéré comme acceptable et non acceptable en termes de probabilité et de conséquences de la condition de défaillance. Comme conséquences acceptables, nous avons des nuisances probables (probabilité inférieure à $1x10^{-5}$), improbables (probabilité entre $1x10^{-5}$ et $1x10^{-9}$), des procédures anormales et d'urgence et des dommages à l'avion, et des effets adverses extrêmement improbables (probabilité inférieure à $1x10^{-9}$) sur occupants et accident catastrophique. Les mêmes conséquences avec une

probabilité supérieure à celles considérées comme acceptées sont considérées comme inacceptables.

2.4.1.7. AMJ 25.1309

Alors que le FAR est développé par FAA (USA), le JAR est développé par JAA (Europe), l'AMJ 25.1309 est la version européenne de l'AC 25.1309.

La structure de l'AMJ 25.1309 est la suivante:

1. Objet

2. Réservé

3. Documents connexes

4. Applicabilité du § / JAR 25.1309

5. Définitions

6. Contexte

7. Classifications des conditions de défaillance et termes de probabilité

8. Objectif de sécurité

9. Conformité avec § / JAR 25.1309

10. Identification des conditions de défaillance et considérations lors de l'évaluation de leurs effets

11. Évaluation des probabilités de condition de défaillance et considérations d'analyse

12. Considérations opérationnelles et d'entretien

13. Évaluation des modifications apportées aux avions précédemment certifiés

Le FAR Part 25 et le JAR-25 ne sont pas totalement similaires, et les différences entre eux peuvent entraîner des coûts supplémentaires pour un avion certifié pour les deux normes. Il est connu que ces coûts supplémentaires ne reflètent pas une augmentation de la sécurité, on pourrait donc considérer qu'une certification est couverte par l'autre.

En tant que FAR Part 25, le JAR-25 est basé sur l'énoncé des moyens de conformité avec la Part 25 1309, en utilisant l'analyse des risques fonctionnels et les évaluations de la sécurité des systèmes.

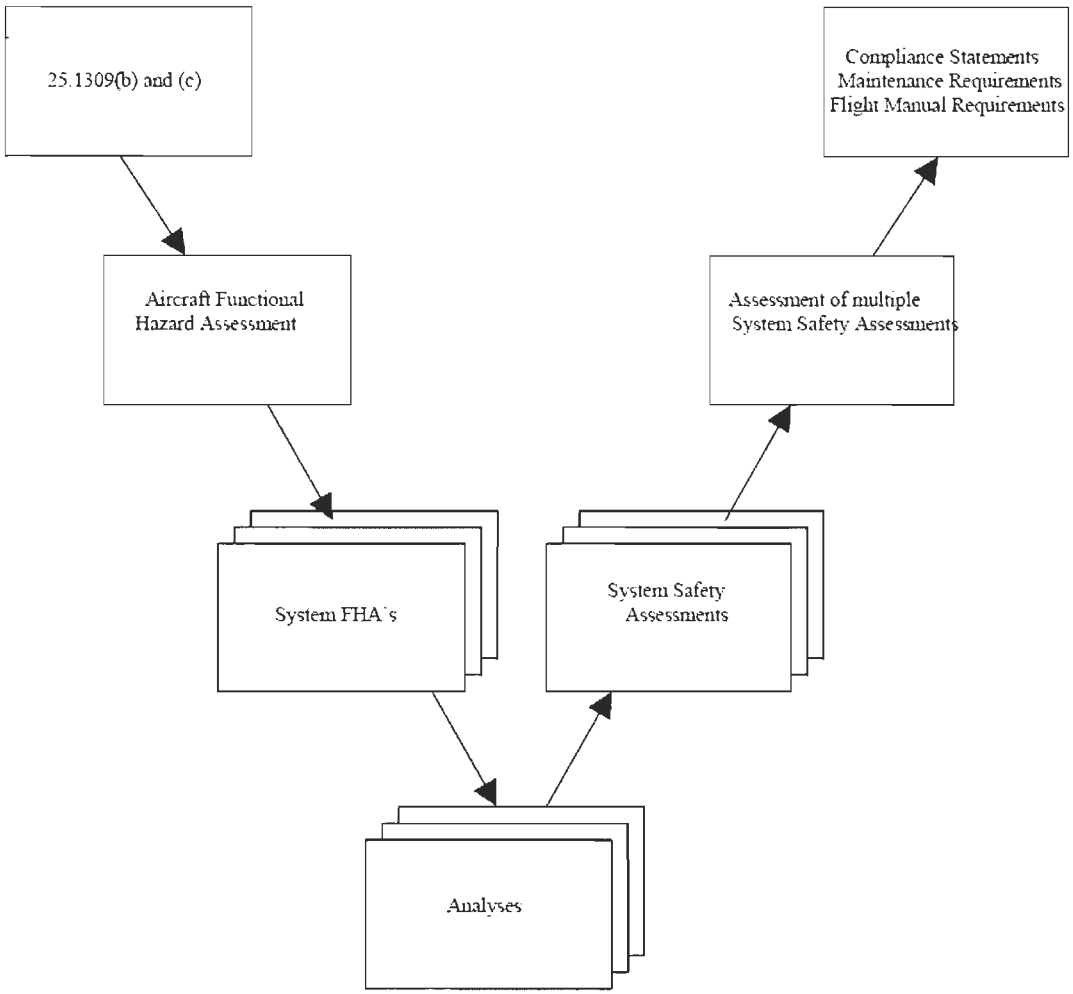

Figure 28 – Safety Assessment selon AMJ 25.1309.

(Crédits d'image: Federal Aviation Administration, 2002)

2.4.1.8. ATA-100

L'ATA-100 est un système de numérotation utilisé principalement pour référencer les systèmes d'aéronefs et les équipements associés. Il est divisé en Avion général (chapitres 00 à 18), Systèmes d'avions (chapitres 20 à 49),

Structure (chapitres 50 à 57), Hélice / Rotor (chapitres 60 à 67), Groupe motopropulseur (chapitres 71 à 84), Divers (chapitres 91 à 116) et Militaire (chapitres 92 à 99).

Par exemple, le système de train d'atterrissage est ATA 32 et le système de commandes de vol est ATA 27. ATA-100 est également divisé en sous-chapitres, comme 27-10 est aileron et tab des commandes de vol.

L'ATA-100 n'est qu'un moyen d'organiser la documentation de l'avion et la société - de nombreuses sociétés sont organisées localement en fonction des numéros ATA-100 en tant que services.

Bien que l'ATA-100 soit très utile, il n'est pas obligatoire de l'utiliser pour passer par le processus de certification.

2.4.1.9. RTCA DO-178

Le but du RTCA DO-178 est de fournir des conseils pour produire des logiciels pour les systèmes et équipements aéroportés qui remplissent leur fonction prévue avec un niveau de confiance en sécurité conforme aux exigences de navigabilité (RTCA DO-178C, 2011). Ces conseils comprennent:

- Objectifs des processus du cycle de vie des logiciels.

- Des activités qui fournissent un moyen de satisfaire ces objectifs.

- Descriptions des preuves sous forme de données de cycle de vie du logiciel qui indiquent que les objectifs ont été atteints.

- Variations des objectifs, de l'indépendance, des données du cycle de vie du logiciel et des catégories de contrôle par niveau de logiciel.

- Considérations supplémentaires (par exemple, les logiciels développés précédemment) applicables à certaines applications.

L'organisation du document est (RTCA DO-178C, 2011):

- Aspects système liés au développement de logiciels
- Cycle de vie du logiciel
- Processus de planification des logiciels
- Processus de développement logiciel
- Processus de vérification du logiciel
- Processus de gestion de la configuration logicielle
- Processus d'assurance qualité des logiciels
- Processus de certification
- Données sur le cycle de vie du logiciel

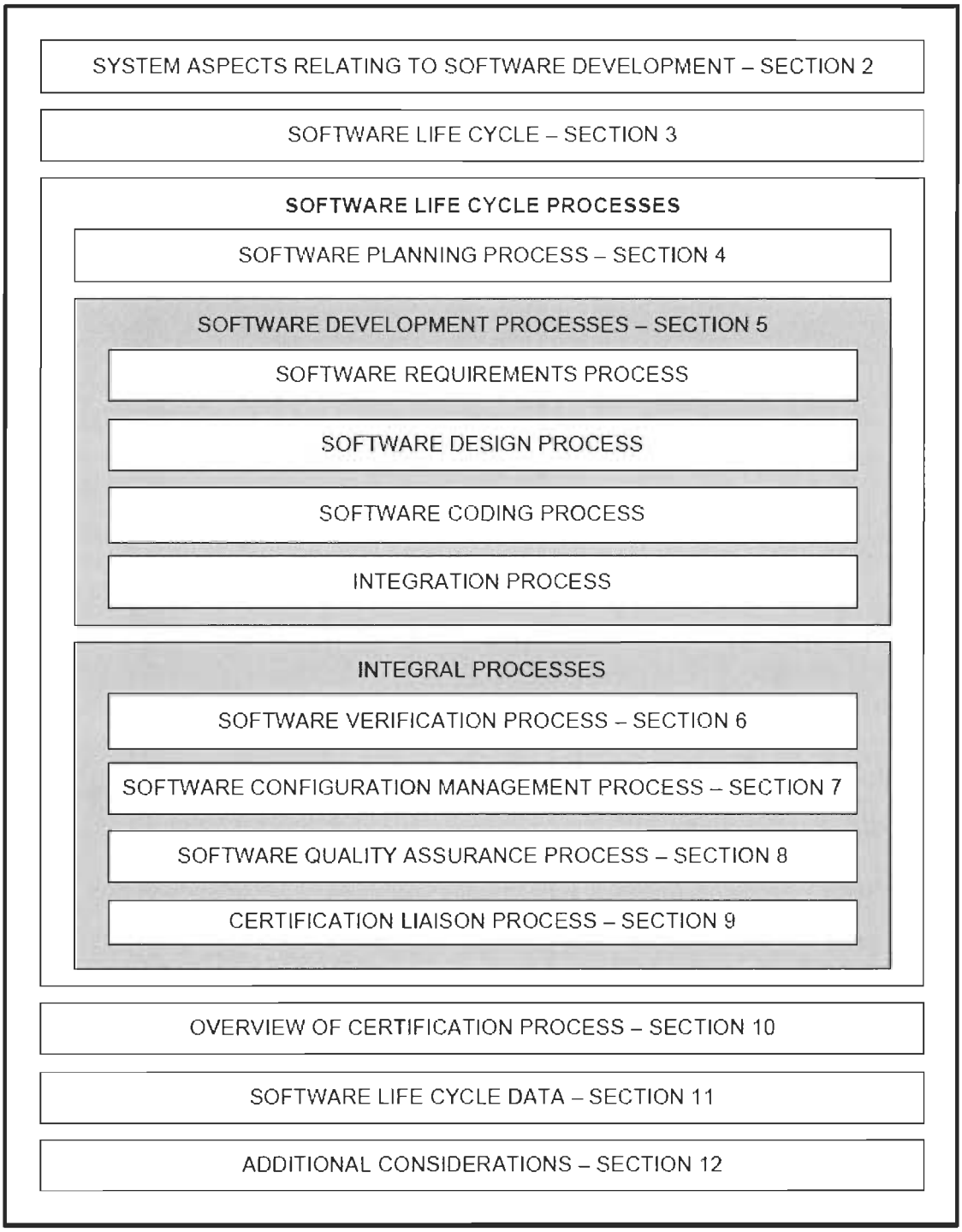

Figure 29 – Aperçu du document RTCA DO-178C

(Crédits d'image: RTCA DO-178C, 2011)

Conditions de défaillance

Le RTCA DO-178 définit les conditions de défaillance (RTCA DO-178C, 2011):

- **Défaillances catastrophiques:** conditions de défaillance qui entraîneraient de multiples décès, généralement avec la perte de l'avion.
- **Défaillances dangereuses:** conditions de défaillance, qui réduiraient la capacité de l'aéronef ou la capacité de l'équipe de conduite à affronter des conditions d'opération défavorables dans la mesure où il y aurait:
    - Une réduction considérable aux marges de sécurité et des capacités fonctionnelles;
    - Détresse physique ou charge de travail excessive telle que l'équipage de conduite ne peut pas être invoqué pour effectuer ses tâches avec précision ou complètement, ou
    - Blessures graves ou mortelles à un nombre relativement restreint d'occupants autres que l'équipage de conduite.

- **Défaillances majeures:** conditions de défaillance qui réduiraient la capacité de l'aéronef ou l'habilité de l'équipe à affronter des conditions d'opération défavorables dans la mesure où il y aurait, par exemple,

- une réduction significative des marges de sécurité ou des capacités fonctionnelles,
- une augmentation significative de la charge de travail de l'équipage ou dans des conditions nuisant à l'efficacité de l'équipage, ou
- inconfort pour l'équipage de conduite, ou
- détresse physique des occupants ou de l'équipe de cabine, y compris éventuellement des blessures.

- **Défaillances mineures:** conditions de défaillance qui ne réduiraient pas de manière significative la sécurité de l'avion et qui impliquent des actions de l'équipage qui sont bien dans la limite de leurs capacités. Les conditions de défaillance mineure peuvent inclure, par exemple,
  - une légère réduction des marges de sécurité ou des capacités fonctionnelles,
  - une légère augmentation de la charge de travail de l'équipage, comme des changements de plan de vol de routine, ou
  - un certain inconfort physique pour les passagers ou le personnel de cabine.
- **Aucune défaillance de l'effet de sécurité:** conditions de défaillance qui n'auraient aucun effet sur la sécurité; par exemple, des conditions de défaillance qui n'affecteraient pas la capacité opérationnelle de l'avion ni augmenteraient la charge de travail de l'équipage.

Définition du niveau logiciel

Le RTCA DO-178 définit cinq niveaux logiciels - A, B, C, D et E - il existe une relation entre ces niveaux et les conditions de défaillance comme suit (RTCA DO-178C, 2011):

Niveau A: logiciel dont le comportement anormal, comme le montre le processus d'évaluation de la sécurité du système (System Safety Assessment), entraînerait ou contribuerait à une défaillance du fonctionnement du système, entraînant une condition de défaillance catastrophique pour l'aéronef.

Niveau B: logiciel dont le comportement anormal, comme le montre le processus d'évaluation de la sécurité du système, entraînerait ou contribuerait à une défaillance du fonctionnement du système entraînant une condition de défaillance dangereuse pour l'aéronef.

Niveau C: logiciel dont le comportement anormal, comme le montre le processus d'évaluation de la sécurité du système, entraînerait ou contribuerait à une défaillance du fonctionnement du système entraînant une condition de défaillance majeure pour l'aéronef.

Niveau D: logiciel dont le comportement anormal, tel que démontré par le processus d'évaluation de la sécurité du système, entraînerait ou contribuerait à une défaillance du fonctionnement du système entraînant une condition de défaillance mineure pour l'aéronef.

Niveau E: logiciel dont le comportement anormal, comme le montre le processus d'évaluation de la sécurité du système, entraînerait ou contribuerait à une défaillance du fonctionnement du système sans effet sur la capacité opérationnelle de l'aéronef ou la charge de travail du pilote. Si un composant logiciel est déterminé comme étant de niveau E et que cela est confirmé par l'autorité de certification, aucune autre directive contenue dans ce document ne s'applique.

Bien que le DO-178 définisse les niveaux du logiciel et sa criticité correspondante, c'est le processus d'évaluation de la sécurité du système qui détermine le niveau de criticité des composants logiciels, en fonction du niveau de gravité de la condition de défaillance du système causée par une défaillance de ce composant logiciel.

Niveau logiciel et objectifs des processus

Le RTCA DO-178 a une liste d'objectifs de processus à atteindre, et le niveau logiciel est directement défini par un ensemble d'objectifs de processus satisfaits pendant le développement logiciel.

Non seulement le nombre d'objectifs de processus doit être atteint dans le développement logiciel, mais également les objectifs de processus qui doivent être satisfaits pour être considérés comme un niveau logiciel déterminé.

Il existe neuf processus de développement logiciel avec des objectifs à satisfaire; chaque processus a ses objectifs propres et différents; le nombre total d'objectifs est de 71.

Logiciel niveau A

Le niveau de logiciel A doit satisfaire tous les objectifs, soixante et onze, et certains de ces objectifs doivent être satisfaits avec indépendance. Les tableaux de l'annexe A du DO-178C indiquent quels objectifs doivent être satisfaits en toute indépendance. Le nombre d'objectifs à satisfaire avec indépendance du niveau logiciel A est de trente.

Il existe deux types d'indépendance. L'indépendance du processus de vérification du logiciel signifie que les activités de vérification doivent être effectuées par une personne ou une équipe différente de la personne ou de l'équipe responsable des activités de développement. Pour le processus d'assurance de la qualité, l'indépendance signifie également inclure le pouvoir d'assurer des mesures correctives.

Tableau 4 - Relation entre le niveau logiciel et le nombre d'objectifs

Niveau DAL	Failure Condition	Nombre d'objectifs à atteindre	Nombre d'objectifs à satisfaire avec l'indépendance
A	Catastrophique	71	30
B	Hazardous	69	18
C	Major	62	5
D	Minor	26	2
E	No effect	0	0

Logiciel niveau B

Les objectifs à atteindre pour le logiciel de niveau B sont définis dans l'annexe A du DO-178. Ils sont soixante-neuf, seulement deux de moins que les logiciels de niveau A. Le nombre d'objectifs à satisfaire avec indépendance est de dix-huit.

Logiciel niveau C

Les objectifs à atteindre pour le logiciel de niveau C sont définis dans l'annexe A du DO-178. Ils sont soixante-deux. Le nombre d'objectifs à satisfaire avec indépendance est de cinq.

Logiciel niveau D

Les objectifs à atteindre pour le logiciel de niveau D sont définis dans l'annexe A du DO-178. Ils ont vingt-six ans. Le nombre d'objectifs à satisfaire avec l'indépendance est de deux.

Logiciel niveau E

Il n'y a aucun objectif à satisfaire pour le logiciel de niveau E.

Processus Logiciels

Les processus avec le nombre d'objectifs connexes sont les suivants (RTCA DO-178C, 2011):

1. Processus de Planification du Logiciel (sept objectifs)
2. Processus des Exigences Logicielles (deux objectifs)
3. Processus de Conception de Logiciels (trois objectifs)
4. Processus de Codage Logiciel (un objectif)
5. Processus d'Intégration (un objectif)
6. Processus de vérification du logiciel (quarante-trois objectifs)
7. Processus de gestion de la configuration logicielle (six objectifs)
8. Processus d'assurance qualité des logiciels (cinq objectifs)
9. Processus de liaison de certification (trois objectifs)

Objectifs du Processus de Planification du Logiciel

Selon la DO-178, le Processus de Planification du Logiciel a sept objectifs (RTCA DO-178C, 2011):

1. Les activités des procès de développement de logiciel et des procès intégraux du cycle de vie logiciel sont définies.

2. Le ou les cycles de vie du logiciel, y compris les interrelations entre les processus, leur séquençage, les mécanismes de feedback et les critères de transition sont déterminés.

3. L'environnement du cycle de vie logiciel a été sélectionné et défini.

4. Des considérations supplémentaires, telles que celles abordées à la section 12, ont été traitées, si nécessaire.

5. Des normes de développement logiciel conformes aux objectifs de sécurité du système pour le logiciel à produire sont définies.

6. Des plans logiciels conformes aux sections 4.3 et 11 ont été produits.

7. L'élaboration et la révision des plans logiciels sont coordonnées.

Objectifs du Processus des Exigences Logicielles

Selon la DO-178, le Processus des Exigences Logicielles a deux objectifs (RTCA DO-178C, 2011):

1. Les besoins de haut niveau sont définis.

2. Les besoins dérivées de haut niveau sont définis et fournis aux processus du système, y compris le processus d'évaluation de la sécurité du système.

Objectifs du Processus de Conception de Logiciels

Le Processus de Conception de Logiciels a trois objectifs (RTCA DO-178C, 2011):

1. L'architecture logicielle est développée.

2. Les exigences de bas niveau sont développées à partir des exigences de haut niveau.

3. Les exigences de bas niveau dérivées sont définies et fournies aux processus du système, y compris le processus d'évaluation de la sécurité du système.

Objectif du Processus de Codage Logiciel

Le Processus de Codage Logiciel a un objectif (RTCA DO-178C, 2011):

1. Le code source est développé à partir des exigences de bas niveau.

Objectif du Processus d'Intégration

Le Processus d'Intégration a un objectif (RTCA DO-178C, 2011):

1. Le code objet exécutable et ses fichiers d'éléments de données de paramètres associés, le cas échéant, sont produits et chargés dans le hardware cible pour l'intégration hardware/logicielle.

Objectifs du Processus de Vérification du Logiciel

Le Processus de Vérification du Logiciel a quarante-trois objectifs (RTCA DO-178C, 2011):

1. Les exigences de haut niveau sont conformes aux exigences du système.

2. Les exigences de haut niveau sont précises et cohérentes.

3. Les exigences de haut niveau sont compatibles avec l'ordinateur cible.

4. Les exigences de haut niveau sont vérifiables.

5. Les exigences de haut niveau sont conformes aux normes.

6. Les exigences de haut niveau sont traçables aux exigences du système.

7. Les algorithmes de haut niveau sont précis.

8. Les exigences de bas niveau sont conformes aux exigences de haut niveau.

9. Les exigences de bas niveau sont précises et cohérentes.

10. Les exigences de bas niveau sont compatibles avec l'ordinateur cible.

11. Les exigences de bas niveau sont vérifiables.

12. Les exigences de bas niveau sont conformes aux normes.

13. Les exigences de bas niveau sont liées à des exigences de haut niveau.

14. Les algorithmes de bas niveau sont précis.

15. L'architecture logicielle est compatible avec les exigences de haut niveau.

16. L'architecture logicielle est cohérente.

17. L'architecture logicielle est compatible avec l'ordinateur cible.

18. L'architecture logicielle est vérifiable.

19. L'architecture logicielle est conforme aux normes.

20. L'intégrité du partitionnement logiciel est confirmée.

21. Le code source est conforme aux exigences de bas niveau.

22. Le code source est conforme à l'architecture logicielle.

23. Le code source est vérifiable.

24. Le code source est conforme aux normes.

25. Le code source est traçable aux exigences de bas niveau.

26. Le code source est précis et cohérent.

27. La sortie du processus d'intégration logicielle est complète et correcte.

28. Le fichier d'élément de données de paramètre est correct et complet

29. La vérification du fichier des éléments de données des paramètres est atteinte.

30. Le code objet exécutable est conforme aux exigences de haut niveau.

31. Le code objet exécutable est robuste avec des exigences de haut niveau.

32. Le code objet exécutable est conforme aux exigences de bas niveau.

33. Le code objet exécutable est robuste avec des exigences de bas niveau.

34. Le code objet exécutable est compatible avec l'ordinateur cible.

35. Les procédures de test sont correctes.

36. Les résultats des tests sont corrects et les écarts expliqués.

37. La couverture des tests des exigences de haut niveau est atteinte.

38. La couverture des tests des exigences de bas niveau est atteinte.

39. La couverture des tests de la structure du logiciel (condition modifiée / couverture de décision) est atteinte.

40. La couverture des tests de la structure du logiciel (couverture des décisions) est atteinte.

41. La couverture des tests de la structure du logiciel (couverture des déclarations) est atteinte.

42. La couverture des tests de la structure du logiciel (couplage de données et couplage de contrôle) est atteinte.

43. La vérification du code supplémentaire, qui ne peut pas être retracée au code source, est réalisée.

Objectifs du Processus de Gestion de la Configuration Logicielle

Le Processus de Gestion de la Configuration Logicielle a neuf objectifs (RTCA DO-178C, 2011):

1. Chaque élément de configuration et ses versions successives sont étiquetés sans ambiguïté afin qu'une base soit établie pour le contrôle et la référence des éléments de configuration.

2. Les bases de référence sont définies pour une activité de processus de cycle de vie logicielle supplémentaire et permettent la référence, le contrôle et la traçabilité entre les éléments de configuration.

3. Le processus de signalement des problèmes enregistre le non-respect des plans et des normes logiciels, enregistre les lacunes des résultats des

processus du cycle de vie des logiciels, enregistre le comportement anormal des produits logiciels et assure la résolution de ces problèmes.

4. Le contrôle des modifications permet l'enregistrement, l'évaluation, la résolution et l'approbation des modifications tout au long du cycle de vie du logiciel.

5. L'examen des modifications garantit que les problèmes et les modifications sont évalués, approuvés ou refusés, que les modifications approuvées sont mises en œuvre et que des commentaires sont fournis aux processus concernés par le biais de rapports de problèmes et de méthodes de contrôle des modifications définies au cours du processus de planification du logiciel.

6. La comptabilité d'état fournit des données pour la gestion de la configuration des processus du cycle de vie du logiciel en ce qui concerne l'identification de la configuration, les références, les rapports de problème et le contrôle des modifications.

7. L'archivage et la récupération garantissent que les données du cycle de vie du logiciel associées au produit logiciel peuvent être récupérées en cas de besoin de dupliquer, régénérer, retester ou modifier le produit logiciel. L'objectif de l'activité de publication est de garantir que seuls les logiciels autorisés sont utilisés, en particulier pour la fabrication de logiciels, en plus d'être archivés et récupérables.

8. Le contrôle de la charge logicielle garantit que le code d'objet exécutable et les fichiers d'éléments de données de paramètres, le cas échéant, sont chargés dans le système ou l'équipement avec les protections appropriées.

9. Le contrôle de l'environnement du cycle de vie du logiciel garantit que les outils utilisés pour produire le logiciel sont identifiés, contrôlés et récupérables.

Objectifs du Processus d'Assurance Qualité des Logiciels

Le Processus d'Assurance Qualité des Logiciels a cinq objectifs (RTCA DO-178C, 2011):

1. L'assurance est obtenue que les plans et les normes de logiciel sont développés et examinés pour la conformité avec ce document et pour la cohérence.

2. L'assurance est obtenue que les processus du cycle de vie des logiciels sont conformes aux plans logiciels approuvés.

3. L'assurance est obtenue que les processus du cycle de vie des logiciels sont conformes aux normes logicielles approuvées.

4. L'assurance est obtenue que les critères de transition pour les processus du cycle de vie du logiciel sont satisfaits.

5. L'assurance est obtenue que la révision de la conformité du logiciel est effectuée.

Objectifs du Processus de Liaison de Certification

Le Processus de Liaison de Certification a trois objectifs (RTCA DO-178C, 2011):

1. La communication et la compréhension entre le postulant et l'autorité de certification sont établies.

2. Les moyens de conformité sont proposés et un accord avec le plan pour les aspects logiciels de la certification (PSAC) est obtenu.

3. Une justification de la conformité est fournie.

DO-178 - Cycle de vie du logiciel

En parlant du cycle de vie du logiciel, DO-178C dit qu'il est habituel d'avoir une séquence des phases suivantes: exigences, conception, codage et intégration. Il donne quelques exemples de différents cycles de vie possibles, dit que l'itération est habituelle et même l'utilisation de prototypage.

Il est présenté quatre types de cycles de vie, chacun pour un composant différent nommé X, Y, Z et W.

Le composant logiciel W est développé à la suite d'une seule itération d'exigences, de conception, de codage et d'intégration (R-D-C-I).

L'exemple du composant X est un logiciel développé précédemment, il n'a donc pas besoin de conception ni de codage, uniquement des exigences et de l'intégration.

L'exemple de composant logiciel Y n'a pas besoin d'être conçu car il est simple et le code peut être produit directement à partir des exigences.

Le composant logiciel de l'exemple Z utilise le prototypage. Le prototype passe par deux cycles de codage-intégration permettant d'affiner les exigences et permettant une meilleure conception avant le codage et l'intégration finaux.

Les logiciels aéroportés sont généralement complexes et de nombreuses itérations peuvent être nécessaires pour obtenir le produit final.

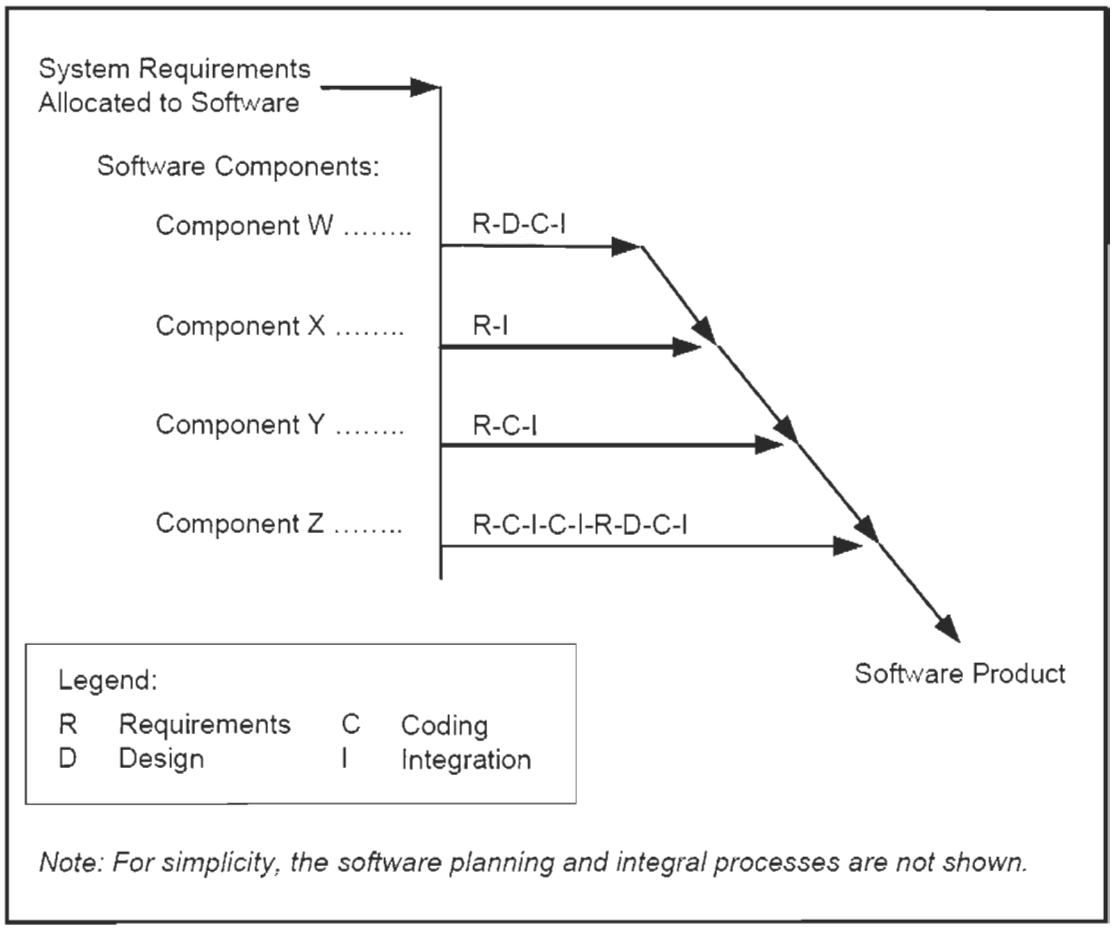

Figure 30 – Exemple de cycle de vie logiciel de DO-178C

(Crédits d'image: RTCA DO-178C, 2011)

Bien que le DO-178C ne dise rien sur les Méthodes Agiles, la présence de l'itération et du prototypage montre que les Méthodes Agiles peuvent être utilisées pour développer des logiciels certifiés DO-178C aéroportés.

Données du cycle de vie du logiciel

Les données du cycle de vie du logiciel sont l'ensemble des données produites pendant le cycle de vie du logiciel utilisées pour planifier, diriger, expliquer, définir, enregistrer ou fournir des preuves d'activités (RTCA DO-178C, 2011). Les caractéristiques des données du cycle de vie du logiciel sont que les données doivent être: non ambiguës, complètes, vérifiables, cohérentes, modifiables et traçables (RTCA DO-178C, 2011).

Les données du cycle de vie du logiciel se composent principalement de documents, de rapports et de code. Cela consiste en (RTCA DO-178C, 2011):

Plan pour les aspects logiciels de la certification (PSAC)

Plan de développement logiciel

Plan de vérification des logiciels

Plan de gestion de la configuration logicielle

Plan d'assurance qualité des logiciels

Normes des exigences logicielles

Normes de conception de logiciels

Normes de code logiciel

Données des exigences logicielles

Description de la conception

Code source

Code d'objet exécutable

Cas et procédures de vérification des logiciels

Résultats de la vérification du logiciel

Index de configuration de l'environnement du cycle de vie du logiciel

Index de configuration logicielle

Rapports de problèmes

Enregistrements de gestion de la configuration logicielle

Dossiers d'assurance qualité des logiciels

Résumé des réalisations du logiciel

Données de trace

Fichier d'élément de données de paramètre

2.4.1.10. RTCA DO-254

Alors que le DO-178 est une norme pour fournir des conseils pour le développement de logiciels, le DO-254 fournit des conseils pour le développement de hardware électronique aéroporté, afin que ce matériel remplisse en toute sécurité sa fonction prévue, dans ses environnements spécifiés (DO-254, 2000). Ces conseils comprennent:

- Définition des objectifs d'assurance de la conception du hardware.

- Décrire la base de ces objectifs pour garantir une interprétation correcte des orientations.

- Fournir une description des objectifs pour permettre le développement de moyens de conformité avec ce guide et d'autres.

- Fournir des conseils pour les activités d'assurance de conception afin d'atteindre les objectifs d'assurance de conception.

- Permettre une flexibilité dans le choix des processus nécessaires pour atteindre les objectifs de ce document, y compris des améliorations, à mesure que de nouvelles technologies de processus deviennent disponibles.

L'organisation du document est (DO-254, 2000):

- Aspects système de l'assurance de la conception du hardware
- Cycle de vie de la conception hardware
- Processus de planification
- Processus de conception du hardware
- Processus de validation et de vérification
- Processus de gestion de la configuration
- Assurance des processus
- Processus de liaison de certification
- Données sur le cycle de vie de la conception du hardware
- Considérations supplémentaires

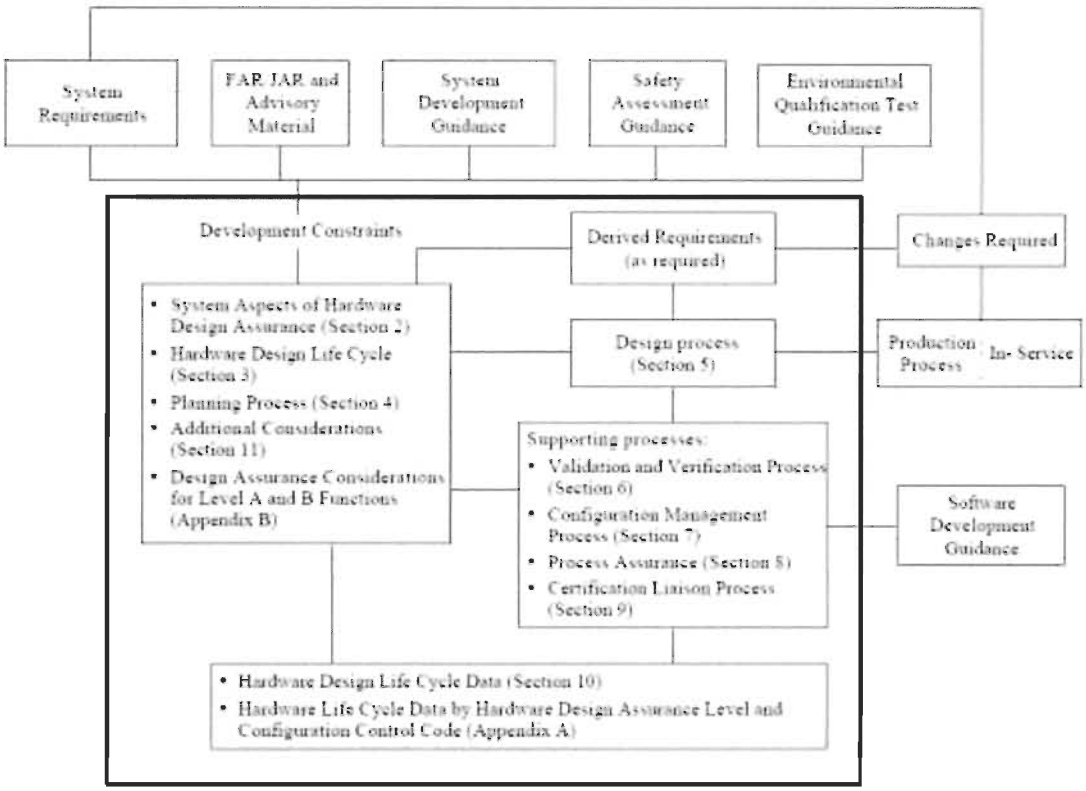

Figure 31 – Aperçu du document RTCA DO-254

(Crédits d'image RTCA DO-254, 2000)

Conditions de défaillance et niveaux d'assurance de conception hardware

Le RTCA DO-254 a la même classification de condition de défaillance et les mêmes définitions de niveau d'assurance de conception hardware du DO-178 pour les logiciels.

Comme dans le DO178, le hardware doit être développé en fonction du niveau de criticité que la défaillance du système hardware cause à l'avion sur lequel il est installé.

Objectifs du niveau hardware et processus

Comme dans le DO-178, il existe une liste d'objectifs à atteindre pendant le développement du hardware, et le niveau d'assurance de la conception du hardware est défini par lequel les objectifs doivent être satisfaits pour atteindre le niveau souhaité.

La catégorie de contrôle hardware est définie et est nécessaire pour décrire comment les objectifs DAL du hardware doivent être satisfaits. Les deux catégories de contrôle hardware existantes, le contrôle hardware 1 et 2 (HC1 et HC2) sont liées à la gestion de la configuration. Le HC1 nécessite que toutes les activités de gestion de configuration soient effectuées et le HC2 nécessite moins d'activités.

Les activités de configuration sont (DO-254, 2000):

- Identification de la configuration
- Lignes de base
- Traçabilité de ligne de base
- Rapport de problème
- Contrôle des changements - intégrité et identification
- Contrôle des changements - enregistrements, approbations et traçabilité
- Libération
- Récupération
- La conservation des données

- Protection contre les modifications non autorisées

- Sélection de médias, rafraîchissement, duplication

Les activités de configuration nécessaires pour le HC2 sont (DO-254, 2000):

- Identification de la configuration

- Traçabilité de ligne de base

- Contrôle des changements - intégrité et identification

- Récupération

- La conservation des données

- Protection contre les modifications non autorisées

Les données du cycle de vie du hardware contiennent de nombreux éléments, comprenant la documentation, les procédures et les rapports de résultats et les dessins. Chaque élément de la liste des données du cycle de vie du hardware a des objectifs associés à satisfaire avec HC1 ou HC2 selon le DAL nécessaire. La liste des données du cycle de vie du hardware est la suivante (DO-254, 2000):

- Plan des aspects de hardware de la certification (PHAC - Plan for Hardware Aspects of Certification)

- Plan de conception du hardware

- Plan de validation du hardware

- Plan de vérification du hardware

- Plan de gestion de la configuration hardware
- Plan d'assurance du processus hardware
- Normes de conception du hardware
- Normes d'exigences
- Normes de conception du hardware
- Normes de validation et de vérification
- Normes d'archivage du hardware
- Données de conception hardware
- Exigences du hardware
- Données de représentation de la conception hardware
- Données de conception conceptuelle
- Données de conception détaillées
- Dessin de haut niveau
- Dessins d'assemblage
- Dessins de contrôle d'installation
- Données d'interface hardware / logicielle
- Données de validation et de vérification
- Données de traçabilité hardware
- Procédures de revue et d'analyse du hardware
- Résultats de revue et d'analyse du hardware
- Procédures de test du hardware
- Résultats des tests hardwares

- Critères de test d'acceptation du hardware

- Rapports de problèmes

- Enregistrements de gestion de la configuration hardware

- Dossiers d'assurance du processus hardware

- Résumé de l'accomplissement du hardware

À titre d'exemple, le Plan des aspects de hardware de la certification a les objectifs suivants (DO-254, 2000):

1. Les processus du cycle de vie de la conception de hardware sont définis.

2. Les normes sont sélectionnées et définies.

3. Les environnements de développement et de vérification du hardware sont sélectionnés ou définis.

4. Les moyens de conformité des objectifs d'assurance de la conception du hardware, y compris les stratégies identifiées à l'aide des directives de la section 2.3.4 (du DO-254), sont proposés à l'autorité de certification.

Dans cet exemple, tous les niveaux A à D ont les objectifs à satisfaire avec HC1.

Les objectifs du plan de conception du hardware sont satisfaits de HC2 du niveau A à C et pour le niveau D, il n'est pas nécessaire de le satisfaire.

La liste complète des objectifs pour chaque élément des données du cycle de vie du hardware se trouve dans le DO-254.

DO-254 - Cycle de vie du hardware

Le DO-178 a décrit le cycle de vie du logiciel comme une séquence de phases, à savoir les exigences, la conception, le codage et l'intégration. Ce document indique également que l'itération est habituelle dans le processus de développement logiciel.

Le DO-254 décrit également le cycle de vie du hardware comme une séquence de phases, à savoir la capture des exigences, la conception, la conception détaillée, la mise en œuvre et la transition de la production. Ce document décrit chaque processus, ses objectifs et les activités connexes qui devraient être abordés pour réduire la probabilité d'erreurs de conception et de mise en œuvre qui affectent la sécurité (DO-254).

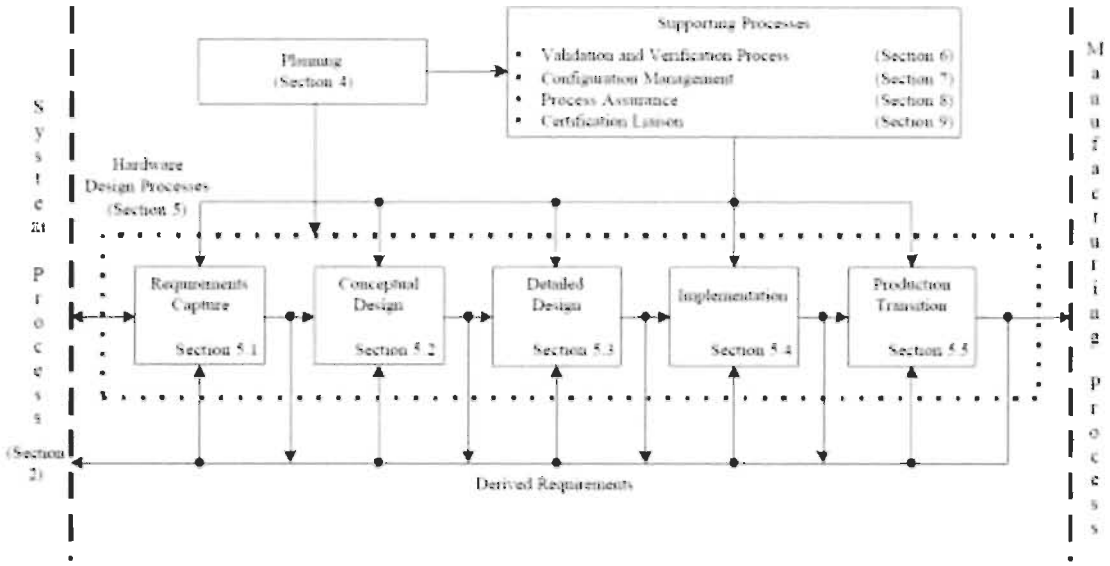

Figure 32 – Cycle de vie du hardware (DO-254)

(Crédits d'image RTCA DO-254, 2000)

Le DO-254 indique également que chaque processus et les interactions entre les processus peuvent être itératifs et, pour chaque itération, l'effet du changement sur chacun des processus doit être pris en compte et évalué afin de déterminer son impact sur les résultats des itérations précédentes. Comme nous pouvons le voir dans la *Figure 32*, le flux d'itération possible est via la ligne des exigences dérivées (Derived Requirements), en d'autres termes, à la fin d'une phase, il est possible de rassembler de nouvelles exigences ou de modifier les exigences, à ses propres frais ou impact.

2.4.1.11. RTCA DO-160

Nous pouvons ajouter à cette liste de documentation le RTCA DO-160, conditions environnementales et procédures de test pour les équipements aéroportés (Environmental Conditions and Test Procedures for Airborne Equipment), utilisé pour qualifier l'équipement des aéronefs en fonction des conditions dans lesquelles chaque équipement devrait se trouver dans un emplacement d'installation donné dans l'aéronef. Selon l'emplacement de l'équipement à l'intérieur de l'avion, il peut être testé pour plusieurs éléments et doit être approuvé avant d'être installé à l'intérieur de l'avion. L'équipement doit être repensé et reconstruit au cas où il ne passerait pas un certain test, et soumis au même test jusqu'à ce qu'il soit réussi avant d'être approuvé pour être installé dans l'aéronef. Toutes les unités de production d'un équipement donné ne seront pas soumises aux tests, certaines d'entre elles sont des tests destructifs, mais une fois qu'un échantillon représentatif est testé et que toutes les unités de production sont garanties d'être similaires, il est approuvé pour être installé dans l'avion si le test réussit.

Il existe une longue liste de tests dans le RTCA DO-160, ils sont divisés par sections, et chaque section a un certain nombre de catégories qui définissent le niveau de gravité. Par exemple, la section sur la température et l'altitude a défini 20 catégories de niveaux d'altitude, de 4 600 à 21 300 mètres combinés à différentes conditions de température contrôlées ou non contrôlées. La variation de température n'a que 5 catégories (DO-160, 2020).

Le RTCA DO-160 a défini les sections suivantes (DO-160, 2020):

4.0 - Température et altitude

5.0 - Variation de température

6.0 - Humidité

7.0 - Chocs opérationnels et sécurité en cas de collision

8.0 - Vibration

9.0 - Atmosphère explosive

10.0 - Imperméabilité

11.0 - Sensibilité aux fluides

12.0 - Sable et poussière

13.0 - Résistance aux champignons

14.0 - Brouillard salin

15.0 - Effet magnétique

16.0 - Entrée d'alimentation

17.0 - Pic de tension

18.0 - Susceptibilité conduite par fréquence audio - Entrées d'alimentation

19.0 - Sensibilité induite au signal

20.0 - Susceptibilité aux radiofréquences (rayonnée et conduite)

21.0 - Émission d'énergie radiofréquence

22.0 - Susceptibilité transitoire induite par la foudre

23.0 - Effets directs de la foudre

24.0 - Givrage

25.0 - Décharge électrostatique

26.0 - Feu et inflammabilité

Le DO-160 contient des instructions détaillées sur la façon d'effectuer les tests pour chaque catégorie et a défini la portée et le but du test dans chaque section.

Lorsque le constructeur d'avions doit acheter un équipement auprès de tiers, cela donne au fournisseur les besoins de qualification de l'équipement, en fonction des caractéristiques environnementales de l'emplacement de l'équipement. Il appartient au tiers de fournir soit un équipement COTS (commercial standard) déjà qualifié, soit d'en développer un nouveau conforme à la définition de qualification.

Le DO-160 est un standard utilisé uniquement pendant la phase de test, il ne définit donc rien sur le cycle de vie des composants.

2.4.2. Phases du projet selon PMI

Le Project Management Institute (Project Management Institute, 2017a), divise les projets en phases ou sous-composantes distinctes et donne des exemples de ce que peuvent être ces phases:

- Développement du concept,
- Étude de faisabilité,
- Exigences du client,
- Développement de solutions,
- Conception,
- Prototypage,
- Construction,
- Tests,
- Transition,
- Mise en service,
- Revue des jalons, et
- Retours d'expérience.

2.4.3. Processus de développement des avions Airbus

Le cycle de vie du développement des avions d'Airbus comporte des jalons connus sous le nom de portes de maturité (Maturity Gate - MG) qui viennent de MG1 à MG15 (Airbus, 2017). Ces portes de maturité peuvent être organisées en phase de faisabilité (MG1-MG3), phase de concept (MG3-MG5), phase de conception (MG5-MG7) et phases finales (MG7-MG15).

Figure 33 – Cycle de vie du développement de l'avion et portes de maturité de l'Airbus (MGs)

Lauzier, (Lauzier 2014), définit les phases finales comme phase d'intégration et de qualification. Il nomme également toutes les MG et ajoute plus d'informations sur le processus de développement d'Airbus. Selon lui, les MG sont:

MG1 - Analyse de scénarios

MG2 - Sélection de scénarios

MG3 - Entrée dans le concept

MG4 - Lancement industriel

MG5 - Fin de concept

MG6 - Début de la production de l'avion de développement (S1)

MG7 - Fin de conception

MG8 - Prêt pour l'assemblage et les tests; Début de l'assemblage du S1; Début de la production du S2

MG9 - Prêt pour la chaîne d'assemblage final S1 FAL (Final Assembly Line)

MG10 - Prêt pour l'allumage du S1

MG11 - Prêt pour le premier vol; Fin de qualification du S1

MG12 - Début de l'assemblage du S2; Début de la production de l'avion de série (S3)

MG13 - Certification de type (Type Certification); Fin de la qualification du S2; Entrée en service

MG14 - Début de l'assemblage de l'avion de série (S3)

MG15 - Fin de qualification du S3

Lauzier donne également des informations supplémentaires sur le processus de développement d'Airbus, comme la sélection des fournisseurs se produit entre les phases MG2 et MG5 et les tests en vol entre MG11 et MG13.

2.4.4. Processus de développement des avions Embraer

Le cycle de vie du développement des avions Embraer comprend les phases suivantes (Ferreira et al., 2013 et Ribeiro, 2017):

- Études conceptuelles (Conceptual Studies - CS)

- Études préliminaires (Preliminary Studies - PS)

- Conception préliminaire (Preliminary Design - PD)

- Phase de définition initiale (Initial Definition Phase - IDP)

- Phase de définition conjointe (Joint Definition Phase - JDP)

- Phase de conception détaillée et de certification (Detailed Design and Certification Phase - DDP)

- Production en série

- Démission (Phase out)

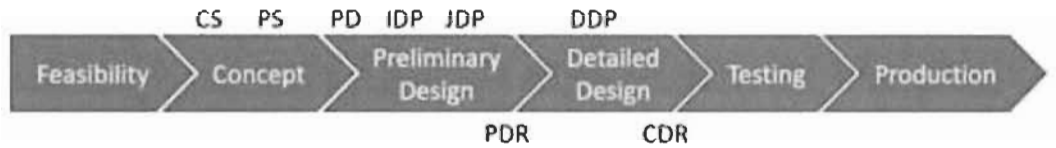

Figure 34 – Cycle de vie du développement Embraer

La phase conceptuelle est divisée en études conceptuelles et études préliminaires. Aux études conceptuelles, il est décrit la solution, basée sur les exigences du marché, définies par le domaine de l'intelligence de marché. La phase d'études conceptuelles nécessite une interaction constante entre la conception préliminaire de l'entreprise et les domaines de l'intelligence de marché. Pendant les études préliminaires commence les processus de sélection et d'embauche des fournisseurs.

La revue de conception préliminaire (PDR) ferme la phase de conception préliminaire et la revue de conception critique (CDR) ferme la phase de conception détaillée.

Au cours de la phase de définition initiale, se produit l'identification des exigences de haut niveau et la définition de la configuration de base.

Pendant la phase de définition conjointe, la validation et les détails du projet initial, y compris la conception, les interfaces et l'assemblage sont définis.

La phase de conception détaillée et de certification est chargée de détailler la conception de toutes les pièces, composants et pièces de l'avion. Des prototypes sont fabriqués, testés et l'avion est certifié.

Dans la production en série, la fabrication, la vente et la livraison d'avions aux clients ont lieu. Des corrections de projet sont apportées à ce stade, en particulier au début.

Au cours de la phase de démission, la planification et le soutien de la fermeture de la ligne de production ont lieu.

2.5. Changement d'exigences

Nous pouvons voir que dans la méthode en cascade, la définition des exigences est une phase fermée et après cela, nous ne pouvons plus changer les exigences. Si nous avons un problème dans la capture des exigences, le plus tard nous trouvons le problème, le plus le coût à corriger sera élevé. Les problèmes liés à l'échec des exigences sont la refonte, la reconstruction, le nouveau test et la refonte de la documentation. L'un des moyens les plus simples d'économiser du temps, de l'argent et des ressources est d'orienter le processus pour éviter de faire les choses plus d'une fois (MacConnell, 1996).

Comme nous l'avons vu dans l'ARP4654A, une norme importante pour l'industrie aéronautique, le processus de développement des avions implique une phase rigide de détermination des exigences. Sans cette détermination des

exigences, nous ne pouvons pas avoir une phase de validation complète des exigences.

La norme militaire MIL-STD-498 reconnaît que les projets n'ont pas les exigences finales et correctes des utilisateurs à ses débuts, mais il s'agit d'une norme logicielle.

Le processus RAD indique que les exigences peuvent être réduites pour s'adapter au timeboxing lorsque le calendrier risque de retarder, mais cela ne semble pas être la réalité de l'industrie aéronautique. Par exemple, nous ne pouvons pas livrer un avion qui convient à moins de passagers ou moins de bagages ou qui atteint une distance plus courte. Vous risquez d'avoir un avion qui n'a pas d'importance à être acheté par quelqu'un. Au lieu de cela, vous pouvez également avoir une liste d'exigences qui ne sont pas obligatoires ou qui sont faciles à modifier sans mettre en danger les performances requises, comme des caractéristiques cosmétiques.

Le processus de développement de logiciels est passé de méthodes en cascade à de nouvelles méthodes où le coût du changement n'a pas tendance à augmenter au fil du temps. Si un produit peut évoluer de manière incrémentielle et itérative, il peut être testé de manière incrémentielle et itérative et livré de manière incrémentielle et itérative. En prenant une application de traitement de texte comme exemple, vous pouvez fournir un produit partiellement développé qui peut être utile à l'utilisateur. Vous n'avez probablement pas encore le

vérificateur d'orthographe, mais vous pouvez toujours taper votre texte, l'enregistrer pour continuer à le taper plus tard ou l'imprimer plus tard.

Pour être juste, la comparaison entre un produit logiciel et un avion nous oblige à avoir un logiciel très coûteux et complexe comme un Business Support System (BSS) ou un Enterprise Resource Planning (ERP). L'effort, le coût et le temps nécessaires pour développer un tel système peuvent être comparables à ceux d'un développement d'avion. Un système logiciel gros et complexe comme celui-ci a évolué au fil des ans comme l'avion a évolué. Les nouveaux systèmes développés dès le départ ont des composants absolument obligatoires et un minimum d'exigences - sans eux, le système est inutile pour les clients, comme un avion sans moteur ou sans ailes. Si des systèmes logiciels complexes comme celui-ci peuvent être développés à l'aide de méthodologies Agiles, un produit complexe comme un avion devrait également, au moins dans une certaine mesure, utiliser certaines techniques issues de méthodologies Agiles. Certains d'entre eux seront discutés dans la section résultats et discussion du présent travail.

Le développement durable utilise le moins d'énergie possible pendant le développement, et toute correction de projet, y compris les changements d'exigences, affecte négativement la durabilité du projet.

Les modifications des exigences affectent les performances de développement de la même manière que toute modification inattendue affecte les performances de développement. Un changement inattendu affecte

négativement les performances de développement et doit être évité. Trouver des problèmes lors des tests n'est pas souhaité mais inévitable, le budget du projet doit donc en tenir compte depuis le début. Lorsqu'un concept, un design ou un produit doit être corrigé, ce changement affecte négativement le projet. Le projet doit considérer et autoriser les modifications à l'intérieur du budget du projet.

Le changement des exigences dans le processus de développement des avions a un impact négatif sur le projet, entraînant des retards et une augmentation des coûts. Les exigences sont principalement définies dans les phases de faisabilité et de conception. Les problèmes dans la définition des exigences doivent être trouvés autant que possible dans les phases de faisabilité et de conception, car le coût de trouver des problèmes d'exigences dans les phases suivantes est connu pour être plus cher.

2.6. Performance de développement

La mesure de la performance est un processus lorsque des informations concernant la performance d'un système ou d'un composant sont collectées ou analysées (Upadhaya et al., 2014).

La mesure de la performance peut également être considérée comme "le processus de quantification de l'efficience et de l'efficacité des actions passées" (Neely et al., 2002) ou "le processus d'évaluation de la qualité de la gestion des organisations et de la valeur qu'elles apportent aux clients et aux autres parties prenantes" (Moullin, M. 2002).

Une mesure quantitative de la performance de développement n'est pas dans le cadre de ce travail. Cependant, l'équipe Scrum est auto-évaluée lors du passage d'un sprint à l'autre à l'aide de la liste de contrôle (checklist), tandis que cela aide l'équipe à progresser vers de meilleures performances.

La division du projet en plusieurs phases permet d'évaluer la performance du projet et de prendre les mesures correctives ou préventives nécessaires dans les phases suivantes (Project Management Institute, 2017a). Les processus du groupe des processus de maîtrise sont utilisés pour suivre, examiner et réglementer l'avancement et la performance du projet; identifier les domaines dans lesquels des changements au plan sont nécessaires; et initiez les modifications correspondantes. Le suivi consiste à collecter des données sur la performance du projet, à produire des mesures de performance, à rendre compte et à diffuser des informations sur la performance. Le contrôle consiste à comparer les performances réelles aux performances planifiées, à analyser les écarts, à évaluer les tendances pour améliorer les processus, à évaluer les alternatives possibles et à recommander des actions correctives appropriées si nécessaire. Avec ces processus, la performance du projet est mesurée et analysée à intervalles réguliers, lors d'événements appropriés ou lorsque des conditions d'exception se produisent afin d'identifier et de corriger les écarts par rapport au plan de gestion de projet.

Les processus de groupe de processus de maîtrise sont (Project Management Institute, 2017a):

- Maîtriser le travail du projet
- Maîtriser les changements
- Valider le périmètre
- Maîtriser le périmètre
- Maîtriser l'échéancier
- Maîtriser les coûts
- Maîtriser la qualité
- Maîtriser les ressources
- Maîtriser les communications
- Maîtriser les risques
- Maîtriser les approvisionnements
- Maîtriser l'engagement des parties prenantes

Les approches itératives, Agiles et adaptatives suivent, examinent et régulent les progrès et les performances en gardant un backlog. Le backlog est priorisé par un représentant de l'organisation avec l'aide de l'équipe de projet qui estime et fournit des informations sur les dépendances techniques. Le travail est tiré du haut du backlog pour la prochaine itération en fonction de la priorité de l'entreprise et de la capacité de l'équipe. Les demandes de changement et les rapports de défauts sont évalués par le représentant de l'organisation en consultation avec l'équipe pour une contribution technique et sont hiérarchisés

en conséquence dans le backlog de travail (Project Management Institute, 2017a).

La performance de développement dans l'industrie aéronautique est mesurée et analysée en comparant le calendrier prévu avec le calendrier réel. Un certain retard doit être inclus dans le budget du projet, car l'historique montre que cela se produit toujours, ou le projet risque d'échouer.

3. METHODOLOGIE

Afin de répondre aux questions de recherche (*Tableau 1*), nous avons défini les Méthodes Agiles, la durabilité, le développement de l'avion, le changement d'exigences et la performance de développement à travers la revue de la littérature. Nous avons défini comment ces éléments sont liés entre eux à travers un cadre conceptuel (*Figure 1*). Certaines questions ont déjà reçu une réponse au cours de la revue de la littérature (les questions RQ1, RQ2, RQ3, RQ4, RQ5, RQ6, RQ8, RQ9, RQ10, RQ11 et RQ12) et d'autres questions recevront une réponse au cours de la discussion (les questions RQ6 et RQ7).

Une relation entre les questions de recherche et le cadre conceptuel est établi. Le *Tableau 1* montre une lettre correspondant au concept présent dans le cadre conceptuel *Figure 1* (par exemple « Durabilité » ou « Performance de développement ») et un chiffre correspondant une flèche du cadre conceptuel (une relation entre deux concepts). Cette relation est importante pour montrer

que tous les éléments du cadre conceptuel sont présents et discutés au cours de ce travail.

Avant de définir les méthodes de développement d'avions, les modèles classiques de développement sont analysés.

Les Méthodes Agiles utilisées dans les logiciels et autres industries sont définies, étudiées (section 2.3. Méthodes Agiles) et comparées au développement actuel des avions pour nous donner les informations nécessaires pour suggérer comment les Méthodes Agiles peuvent être appliquées à l'industrie aéronautique.

Lorsqu'une méthode agile est définie, ses pratiques sont nommées et expliquées, de sorte que ces pratiques peuvent être répertoriées dans un tableau pour être utilisées comme une direction de ce qu'il faut suggérer pour être utilisé dans l'industrie aéronautique.

Les pratiques Agiles expliquées lors de la présentation de chaque Méthode Agile sont résumées dans le tableau 3. Avec cela, nous avons une vue grand angle de toutes les pratiques Agiles utilisées dans toutes les Méthodes Agiles présentées dans ce travail. Un tableau similaire, le tableau 5, montrela même liste de pratiques, et indique maintenant s'il est déjà utilisé dans l'industrie aérospatiale ou s'il est recommandé ou non de l'utiliser dans l'industrie aérospatiale.

Ensuite, les méthodes de développement utilisées présentement dans l'industrie aéronautique sont définies, et nous vérifions s'il existe une pratique Agile utilisée dans l'industrie aérospatiale.

Nous étudions également la manière dont les changements d'exigences et les performances de développement sont traités dans l'industrie aérospatiale.

Dans la section résultats et discussion, nous explorons chaque principe du Manifeste Agile afin d'extraire, de chacun, des questions à poser lors du développement de l'avion. Nous faisons cela pour vérifier si la pratique pointée par le principe du Manifeste Agile est appliquée au développement de l'avion. Ce questionnement a pour but de vérifier si la pratique est utilisée et de recommander qu'elle soit utilisée dans le développement de l'avion.

Enfin, nous montrons un tableau qui reprend chaque pratique Agile discutée dans ce travail et indique si chacune d'elles doit être appliquée ou non dans l'industrie aéronautique.

4. RÉSULTATS ET DISCUSSIONS

Les Méthodes Agiles ont été développées à l'origine pour l'industrie du logiciel, mais en raison de leur grand succès, elles ont été étendues à des projets non informatiques (Serrador et al., 2015). Le travail de Serrador a pris un échantillon de données de 1386 projets dans plusieurs industries et pays pour les tester sur l'utilisation des Méthodes Agiles. Il dit que près de 6% du total des projets étaient complètement ou presque complètement Agile, et plus de 65% ont

déclaré avoir une composante Agile ou itérative. Les projets considérés provenaient d'industries telles que la construction, les services financiers, les services publics, le gouvernement, l'éducation, la haute technologie, les télécommunications, la fabrication, les soins de santé, les services professionnels et la vente au détail.

Le Agile Practice Guide indique que les équipes de projet utilisent des approches Agiles dans une variété d'industries au-delà du développement de logiciels, et il existe un large éventail d'outils, de techniques et de cadres d'application; les équipes ont le choix d'approches et de pratiques qui correspondent à leur projet et à la culture organisationnelle afin d'atteindre le résultat souhaité (Project Management Institute, 2017b).

L'utilisation de techniques Agile ne changera pas une méthode rigide en cascade pour qu'elle soit Agile sans changer son cycle de vie. Nous pouvons gagner de plus en plus de temps en utilisant certaines techniques Agiles. Une autre idée consiste à transformer chaque phase de la méthode en cascade en un sprint Scrum et à la rendre Agile. Au moins, nous pouvons passer par les phases avec des contraintes rigides pour passer chaque phase et avoir toujours les avantages Agile à l'intérieur de la phase (*Figure 35*).

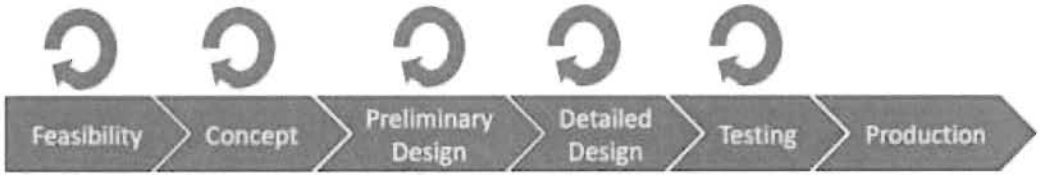

Figure 35 – Développement incrémental et itératif dans chaque phase du projet

Des Méthodes Agiles ont déjà été mises en œuvre avec succès dans la conception conceptuelle d'aéronefs (Glas et al., 2012a; Glas et al., 2012b). Les techniques utilisées étaient la collaboration interdisciplinaire, les cycles d'itération de conception rapides, Scrum, le timeboxing, les réunions quotidiennes, le backlog de produit, les graphiques de burn-down, la collaboration, l'équipe interfonctionnelle (interdisciplinaire), l'intégration précoce, le feedback, la livraison incrémentielle, le développement incrémental, développement itératif, storyboard, rétrospectives de sprint, planification de sprint, développement de courte durée et petites équipes.

Glas déclare que l'industrie aéronautique limite traditionnellement l'utilisation des principes Agiles à la phase de conception. De plus, une solution consisterait à démarrer simultanément toutes les phases du projet, de la conception préliminaire à la vente (Glas et al., 2009).

L'exécution d'une checklist pendant la réunion de la cérémonie de revue est un moyen habituel de vérifier les exigences du processus de développement et

est connue pour être une pratique très courante. L'équipe Scrum a une mentalité construite en connaissant la liste de contrôle au préalable.

Une partie de la solution proposée repose sur l'analyse des méthodes Agiles pour extraire ce qui peut être fait en matière de durabilité. Tous les douze principes du Manifeste Agile seront cités ici pour le comprendre dans le contexte de la durabilité. Le résultat de l'analyse de principe est une question à inclure dans la checklist. Des questions supplémentaires concernant la durabilité du processus de développement des avions sont incluses sur la base des bonnes pratiques des méthodes Agiles.

L'autre partie de la solution proposée est de savoir comment adapter les processus Agiles aux projets aéronautiques de la manière dont ils peuvent être utiles. Dans le développement d'avions, cent pour cent des résultats de toutes les phases, de la faisabilité à la conception détaillée, sont la documentation. L'application des pratiques Agile ne devrait poser aucun problème au cours de ces phases, et pour les phases suivantes où le hardware est impliqué, une certaine adaptation doit être effectuée.

À partir de maintenant, nous travaillons à travers les phases de développement de l'avion en supposant que les concepts de sprint, de backlog de sprint et de rétrospective de sprint sont acceptés pour être utilisés pendant les phases de développement. Chaque principe du Manifeste Agile est cité puis discuté dans la perspective du développement de l'avion.

Premier principe:

'' Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.'' (Beck et al., 2001)

La durabilité de ce principe réside dans les mots "rapidement" et "grande valeur ajoutée". La durabilité maximale est atteinte lorsque l'énergie est dépensée pour fournir les fonctionnalités avec la plus grande valeur possible à la fin d'un sprint.

Les ''fonctionnalités à grande valeur ajoutée'' peuvent être utiles non seulement au logiciel avion mais à l'ensemble du projet aéronautique. Donc, nous parlerons de tout produit livrable que nous pourrions avoir pour le sprint - principalement de la documentation allant de la faisabilité aux phases de conception détaillées.

Plus tôt nous aurons de la valeur à livrer, moins d'énergie sera dépensée pour la développer. L'équipe Scrum doit être suffisamment mature pour dimensionner la plus grande partie du backlog produit à réaliser pendant le sprint.

La question à ajouter à la checklist est: "Le backlog de sprint a-t-il été bien quantifié, afin d'utiliser le plus d'efficacité possible de l'équipe?"

Deuxième principe:

'' Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.'' (Beck et al., 2001)

Les Méthodes Agiles sont un excellent outil pour surmonter le changement des exigences. Un changement dans les exigences ne peut pas compromettre les jalons du projet ni le calendrier. Nous devons nous adapter ici de ''tard dans le projet'' à ''tard dans le sprint''.

La question à ajouter à la checklist est la suivante: "Le rythme, la charge de travail ou le calendrier du projet ont-ils été affectés par les modifications des exigences?"

Troisième principe:

'' Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.'' (Beck et al., 2001)

Ici, nous prenons le mot ''logiciel' comme toute valeur à livrer.

L'itération augmente la durabilité en facilitant la gestion des pièces livrables plus petites.

La question à ajouter à la checklist est: "Le backlog de produit était-il bien dimensionné et partitionné en petits morceaux qui sont bons pour être bien gérés en termes de rythme, de charge de travail et de calendrier?"

Quatrième principe:

'' Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.'' (Beck et al., 2001)

La durabilité est augmentée par le feedback constant des parties prenantes.

La question à ajouter à la checklist est: "Les parties prenantes travaillaient-elles près des développeurs?"

Cinquième principe:

'' Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.'' (Beck et al., 2001)

Il est bon de préférer être un bon leader qu'un patron. La différence est qu'un bon leader sait comment le faire, en plus de savoir quoi faire comme le ferait un patron régulier. Un bon leader maintient toujours l'équipe motivée au lieu de simplement légiférer et réglementer (Tschohl 2015).

La question à ajouter à la checklist est: "L'équipe était-elle suffisamment motivée en ce qui concerne l'environnement et le soutien?"

Sixième principe:

'' La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.'' (Beck et al., 2001)

La communication est un concept clé pour la réussite des projets et les résultats du programme, tout comme elle développe les relations nécessaires. La communication peut avoir les dimensions suivantes (Project Management Institute, 2017a):

• Interne: se concentrer sur les parties prenantes au sein du projet et au sein de l'organisation.

• Externe: Concentrez-vous sur les parties prenantes externes telles que les clients, les fournisseurs, d'autres projets, les organisations, le gouvernement, le public et les défenseurs de l'environnement.

• Formel: Rapports, réunions formelles (à la fois régulières et ad hoc), ordres du jour et procès-verbaux de réunions, briefings des parties prenantes et présentations.

• Informel: activités de communication générales utilisant des courriels, les médias sociaux, des sites Web et des discussions informelles ad hoc.

• Orientation hiérarchique: la position de la partie prenante ou du groupe concernant l'équipe de projet affectera le format et le contenu du message, de la manière suivante:

1. Vers le haut: parties prenantes de la haute direction.
2. Vers le bas: l'équipe et les autres personnes qui contribueront aux travaux du projet.
3. Horizontale: pairs du chef de projet ou de l'équipe.

• Officiel: rapports annuels; rapports aux régulateurs ou aux organismes gouvernementaux.

• Non officiel: communications qui visent à établir et à maintenir le profil et la reconnaissance du projet et à établir des relations solides entre l'équipe du projet et ses parties prenantes en utilisant des moyens flexibles et souvent informels.

• Écrit et oral: verbal (mots et inflexions vocales) et non verbal (langage corporel et actions), médias sociaux et sites Web, communiqués de presse.

La conversation en face à face est considérée comme la meilleure solution pour la communication vers et au sein de l'équipe de développement car cette interaction donne une réponse plus claire et plus rapide aux questions posées. Si une'autre partie n'a pas toutes les réponses pour le moment, il faut noter les questions pour y répondre dès qu'il aura connaissance de ces réponses.

La question à ajouter à la checklist est: "La communication au cours du développement a-t-elle été faite en face à face autant qu'il était possible?"

Il est important de savoir si tous les efforts possibles pour établir une communication en face à face a été faite.

Septième principe:

'' Un logiciel opérationnel est la principale mesure d'avancement.'' (Beck et al., 2001)

Nous pouvons lire sur la page d'accueil du Manifeste Agile que le logiciel opérationnel est plus précieux qu'une documentation exhaustive. Ensuite, après avoir dit cela, il dit que s'il y a de la valeur dans les éléments à droite (''documentation exhaustive'' dans ce cas), il y a plus de valeur dans les éléments à gauche (''logiciel opérationnel'' dans ce cas).

La page d'accueil du Manifeste Agile commence par ceci (Beck et al., 2001):

'' Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

Les individus et leurs interactions plus que les processus et les outils
Des logiciels opérationnels plus qu'une documentation exhaustive
La collaboration avec les clients plus que la négociation contractuelle
L'adaptation au changement plus que le suivi d'un plan

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.'' (Beck et al., 2001)

En d'autres termes, le plus opérationnel nous avons les logiciels, moins nous avons besoin de documentation. Cela ne dit pas que la documentation n'est pas un composant logiciel important.

Cette fois, nous pouvons adapter le mot ''logiciel opérationnel'' à ''la valeur à livrer'' par rapport à l'effort de production de documentation.

Nous avons évidemment un problème ici lorsque la valeur à livrer est la documentation. Nous pouvons essayer de l'adapter au développement des avions ou laisser ce principe derrière. Une tentative d'adaptation consiste à séparer la documentation livrable (liée à la certification) de la documentation non livrable, principalement la documentation bureaucratique.

La question à ajouter à la checklist est: "Quel est le pourcentage de l'effort appliqué pour obtenir la valeur à livrer (documentation liée à la certification) par rapport à l'effort utilisé pour produire l'autre documentation?"

Huitième principe:

''Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.'' (Beck et al., 2001)

Le rythme est très important pour la durabilité du projet. On sait que les heures supplémentaires, les quarts de nuit, les longues périodes de travail et travailler pendant les vacances ont tendance à diminuer l'efficacité. Un temps de productivité faible se traduit par une consommation d'énergie plus élevée.

La question à ajouter à la checklist est: "Le rythme de développement était suffisamment constant pour maintenir le niveau d'efficacité le plus élevé possible?"

Neuvième principe:

'' Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.'' (Beck et al., 2001)

Bien que les méthodes agiles aient tendance à accueillir changement d'exigences (Beck et al., 2001), cela ne signifie pas qu'il n'y a pas de phase de conception dans le projet. Un bon design évite toujours de retravailler le produit. L'attention à l'excellence technique ne doit pas être diminuée au fil du temps pendant le projet.

Deux questions dérivées de ce principe doivent être ajoutées à la checklist:

"Le projet a été retravaillé en raison d'une mauvaise conception?"

"Le niveau technique a-t-il été maintenu élevé tout au long du projet?"

Dixième principe:

'' La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.'' (Beck et al., 2001)

La simplicité dans le développement des avions est obtenue lorsque nous supprimons tous les détails qui ne sont pas essentiels aux fonctionnalités, ainsi que les fonctionnalités qui ne sont pas essentielles à la valeur finale. Il est ici considéré que l'art de décider ce qu'il ne faut pas faire ou ce qui est vraiment essentiel peut être difficile à faire. Nous ajoutons de la valeur au produit final lorsque nous éliminons les éléments de la liste des tâches qui ajoutent moins de valeur et laissons les ressources à utiliser sur ce qui est vraiment important.

La question à ajouter à la checklist est: "L'équipe était-elle capable de séparer les détails (ou fonctionnalités) utiles de manière que les ressources soient utilisées plus intensément?"

Onzième principe:

'' Les meilleures architectures, spécifications et conceptions émergent d'équipes autoorganisées.'' (Beck et al., 2001)

L'une des exigences d'un développement efficace est d'avoir une équipe de développement mature. L'une des qualités d'une équipe mature est d'être auto-organisée. Une équipe de développement mature est capable de dimensionner correctement l'effort de développement, de test et de documentation de chaque User Story. L'équipe de développement mature est également en mesure d'allouer facilement des ressources en fonction du pourcentage de développement, de test et de documentation pour chaque sprint. Différents

backlogs de produits peuvent nécessiter différentes quantités de développement, de test et de documentation, et l'équipe a l'autonomie pour gérer cela.

La question à ajouter à la checklist est: "L'équipe était-elle capable de s'auto-organiser en termes d'allocation des ressources pendant le sprint?"

Douzième principe:

'' À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.'' (Beck et al., 2001)

L'équipe Agile apprend toujours à être plus efficace. Cela se produit en raison de la nature particulière de chaque backlog de produit. Pour maintenir le rythme constant, produire des résultats optimaux et maintenir un haut niveau d'efficacité, l'équipe change et alloue les ressources et affine la manière de produire des avions.

La question à ajouter à la liste de contrôle est: "L'équipe était-elle capable d'ajuster le comportement pour devenir plus efficace en fonction du backlog de produit particulier?"

4.1. Checklist

Voici une liste de questions à poser lors de la cérémonie de revue pour vérifier l'agilité et la durabilité du processus de développement de l'avion. L'équipe doit avoir connaissance de cette liste pendant tous les sprints et le processus se conformera davantage à la liste au fur et à mesure que l'équipe passera de sprint en sprint. Les résultats des sprints précédents doivent être transmis également en cas d'équipes différentes impliquées.

1. Le backlog de sprint a-t-il été bien quantifié, afin d'utiliser le plus d'efficacité possible de l'équipe?
2. Le rythme, la charge de travail ou le calendrier du projet ont-ils été affectés par les modifications des exigences?
3. Le backlog de produit était-il bien dimensionné et partitionné en petits morceaux qui sont bons pour être bien gérés en termes de rythme, de charge de travail et de calendrier?
4. Les parties prenantes travaillaient-elles près des développeurs?
5. L'équipe était-elle suffisamment motivée en ce qui concerne l'environnement et le soutien?
6. La communication au cours du développement a-t-elle été faite en face à face autant qu'il était possible?
7. Quel est le pourcentage de l'effort appliqué pour obtenir la valeur à livrer (documentation liée à la certification) par rapport à l'effort utilisé pour produire l'autre documentation?

8. Le rythme de développement était suffisamment constant pour maintenir le niveau d'efficacité le plus élevé possible?
9. Le projet a été retravaillé en raison d'une mauvaise conception?
10. Le niveau technique a-t-il été maintenu élevé tout au long du projet?
11. L'équipe était-elle capable de séparer les détails (ou fonctionnalités) utiles de manière que les ressources soient utilisées plus intensément?
12. L'équipe était-elle capable de s'auto-organiser en termes d'allocation des ressources pendant le sprint?
13. L'équipe était-elle capable d'ajuster le comportement pour devenir plus efficace en fonction du backlog de produit particulier?

En plus des questions liées au Manifeste Agile énumérées ci-dessus, il existe d'autres questions basées sur les bonnes pratiques de développement logiciel qui peuvent être adaptées au développement de l'avion:

14. Les outils de développement des avions étaient-ils disponibles en tout temps?

15. La dette technique est-elle maintenue à zéro ou la plus faible possible?

16. Le sprint a-t-il été correctement estimé en termes d'effort (temps et ressources)?

17. La documentation est-elle élaborée cornpte tenu de la moindre utilisation possible des ressources?

4.2. Pratiques Agiles recommandées pour l'industrie aéronautique

Nous avons montré une checklist avec les meilleures pratiques Agiles qui ont déjà été appliquées avec succès à l'industrie du logiciel et montré comment une certaine personnalisation peut être effectuée pour les appliquer mieux à l'industrie aéronautique.

Dans la section 2.3.13. (Pratiques des méthodes Agiles) nous avons répertorié les pratiques Agiles les plus utilisées et montré dans quelle méthode Agile chacune d'elles est utilisée.

Maintenant, nous prenons la même liste de pratiques Agile et indiquons celles qui sont déjà utilisées dans l'industrie aéronautique et celles qui sont recommandées, recommandées avec une certaine adaptation, non recommandées ou non applicables. Le résultat est indiqué dans le *Tableau 5*.

Tableau 5 - Pratiques Agiles recommandées pour l'industrie aéronautique

Pratique Agile	Recommandation développement d'avion
Test automatisé	Non applicable
Test manuel	Déjà utilisé
Backlog	Recommandé
Burn-down chart	Recommandé
Clean rooms	Recommandé
Collaboration	Déjà utilisé
Concurrence	Déjà utilisé
Intégration continue	Non applicable
Équipe interfonctionnelle	Déjà utilisé
Réunion quotidienne	Recommandé
Feedback	Recommandé avec adaptation

Livraison incrémentielle	Recommandé avec adaptation
Développement incrémental	Recommandé avec adaptation
Prototypage	Déjà utilisé
Développement itératif	Non recommandé
Kanban	Recommandé
Programmation en binôme	Non applicable
Planning poker	Recommandé
Outils de développement rapide	Non applicable
Retrospective	Recommandé
Développement de courte durée	Recommandé avec adaptation
Petites équipes	Recommandé avec adaptation
Sprint	Recommandé
Planification de sprint	Recommandé
Revue de sprint	Recommandé
Développement piloté par les tests	Non applicable
Timeboxing	Recommandé avec adaptation
Test de l'unité	Non applicable
User story	Recommandé

5. CONCLUSION

La checklist est offerte comme un moyen de vérifier l'agilité et la durabilité des projets aéronautiques, et elle devrait être livrée répondue lors de la cérémonie de rétrospective du sprint.

La façon dont cette checklist est construite, basée sur les meilleures pratiques de développement des avions, permet de fournir un processus de développement d'avions plus durable à l'industrie aéronautique.

Un tableau avec les meilleures pratiques Agile utilisées dans chacune des méthodes Agile les plus utilisées a été présentée et la façon dont chacune de ces pratiques est également recommandée pour l'industrie aéronautique.

Des plus amples recherches permettront de tester la checklist et le tableau proposées, donc des travaux supplémentaires doivent être effectués pour ajuster la checklist et le tableau. Un certain niveau de personnalisation peut être fait pour les adapter aux différents processus de développement des avions.

Des recherches plus approfondies permettront également de mesurer quantitativement le processus de développement pour évaluer une amélioration appropriée du processus de développement en utilisant la checklist et le tableau.

RÉFÉRENCES

14 CFR Part 23 - AIRWORTHINESS STANDARDS: NORMAL, UTILITY, ACROBATIC, AND COMMUTER CATEGORY AIRPLANES. Retrieved March 7, 2020, from https://www.risingup.com/fars/info/23-index.shtml.

14 CFR Part 25 - AIRWORTHINESS STANDARDS: TRANSPORT CATEGORY AIRPLANES. Retrieved March 7, 2020, from https://www.law.cornell.edu/cfr/text/14/part-25.

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. 2002, Agile software development methods: Review and analysis, Espoo 2002, VTT Publications 478.

Abrahamsson, P., Warsta, J., Siponen, M., & Ronkainen, J. (2003). New directions on Agile methods: a comparative analysis. 25th International Conference on Software Engineering, 2003. Proceedings.

Agile Business Consortium (2019), Chapter 4: Principles. (n.d.). Retrieved March 24, 2020, from https://www.Agilebusiness.org/page/ProjectFramework_04_Principles

Airbus. (2017, February 27). Aircraft lifecycle from design to operations. Retrieved November 18, 2019, from https://www.airbus.com/newsroom/news/en/2017/02/aircraft-lifecycle-from-design-to-operations.html.

Albertao, F., 2004, "Sustainable Software Engineering", Masters Practicum Paper, Carnegie Mellon University Sillicon Valley, Moffett Field, California, USA.

ARP4754 Rev. A (2010), Aerospace Recommended Practice, Guidelines for Development of Civil Aircraft and Systems, SAE Aerospace.

ARP4761 (1996), Aerospace Recommended Practice, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, SAE Aerospace.

Baseer, K. K.; Reddy, A. R. M.; Bindu, C. S. A Systematic Survey in Waterfall vs. Agile vs. Lean Process Paradigms" In: Journal on Software Engineering. Jan-Mar2015, Vol. 9 Issue 3, p36. 26p.

Beck K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler,J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, 2001, Manifesto for Agile Software Development, http://Agilemanifesto.org/principles.html

Beck, K., & Andres, C. (2004). Planning Extreme Programming: embrace change. Boston, MA: Addison-Wesley.

Beynon-Davies, P., Carne, C., Mackay, H., and Tudhope, D., 1999, Rapid application development (RAD): an empirical review, European Journal of Information Systems, 8, 211-223.

Blanchard, B. S., & Fabrycky, W. J. (2014). Systems engineering and analysis. Harlow: Pearson Education Limited.

Building, C. I. of. (2014). Code of practice for project management for construction and development (5th ed.). Chichester: John Wiley & Sons, Inc.

Coad, P., Lefebvre, E., & Luca, J. D. (1999). Java modeling in color with UML: enterprise components and process. Upper Saddle River NJ: Prentice Hall PTR.

Cockburn, A., 2000, Selecting a Project's Methodology, IEEE Software, July/August 2000.

Cockburn, A., (2002). Agile Software Development. Boston, Addison-Wesley.

Cockburn, A., 2005, Crystal Clear A Human-Powered Methodology for Small Teams, Addison-Wesley Professional.

Cockburn, A., 2008, Using Both Incremental and Iterative Development, CrossTalk, The Journal of Defense Software Engineering, Vol. 21, No. 5, May 2008, pp. 27-30.

Crouch, T. D. (2019, December 23). Sir George Cayley. Retrieved April 23, 2020, from https://www.britannica.com/biography/Sir-George-Cayley

Dassault Systèmes (July 2010). Simulia website. URL: http://www.simulia.com.

Didion, I. (1837), Rapport sur la plus grande vitesse que l'on peut obtenir par la navigation aérienne, Congrès scientifique de France, 5th session, Metz, p. 548 https://gallica.bnf.fr/ark:/12148/bpt6k411514q/f541

DO-160. (n.d.). Retrieved February 18, 2020, from https://do160.org/rtca-do-160g/

ECSS-M-ST-10C Rev.1, 6 March 2009, Space project management, Project planning and implementation, ECSS Secretariat ESA-ESTEC, Requirements & Standards Division, Noordwijk, The Netherlands.

Eklund U, Bosch J. "Applying Agile development in mass-produced embedded systems". Proc. of Agile Processes in Software Engineering and Extreme Programming, XP 2012. In: Lecture Notes in Business Information Processing 2012; 111: 31–46.

EPA, United States Environmental Protection Agency (2011) " What Exactly Is Sustainability?", http://www.thetomorrowplan.com/exchange/what-exactly-is-sustainability/

Federal Aviation Administration. (1988). AC 25.1309-1A - System Design and Analysis. Retrieved April 28, 2020, from https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/22680

Federal Aviation Administration. (2002). AC/AMJ 25.1309 - System Design and Analysis. Retrieved April 28, 2020, from https://www.faa.gov/regulations_policies/rulemaking/committees/documents/media/TAEsdaT2-5241996.pdf

Fernández, M. (Dec. 2005). "A Framework for Agile Collaboration in Engineering". PhD thesis. School of Mechanical Engineering, Georgia Institute of Technology.

Ferreira, M. J. B.; Sabbatini, R. C. Engenharia de projetos na indústria aeronáutica brasileira. Campinas, SP: Relatório, Unicamp/ABDI, 2013, p.1-42.

Gilb, T. (2003), Evolutionary Project Management (Evo): How to Manage Project Benefits and Costs, INCOSE 2003 - 13th Annual International Symposium Proceedings, pp. 364-373.

Gilb, T., & Brodie, L. (2005). Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage. Oxford: Butterworth-Heinemann, Chapter 10 - EVOLUTIONARY PROJECT MANAGEMENT How to Manage Project Benefits and Costs.

Gilb, T. (2005), 12.2 Fundamental Principles of Evolutionary Project Management. INCOSE International Symposium, vol 15, issue 1, pp. 1733-1742.

Glas, M. and Ziemer, S. (2009). "Challenges for Agile Development of Large Systems in the Aviation Industry". In: Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications. OOPSLA'09. Orlando, Florida, USA: ACM, pp. 901–908.

Glas, M. and Seitz, A., (2012a). "Application of Agile Methods in Conceptual Aircraft Design". In: Deutscher Luft- und Raumfahrtkongress

Glas, M. and Seitz, A. (2012b). In: Survey on Application of Scrum During Phase 1 of Concept 002. Tech. rep. IB 12028. Bauhaus Luftfahrt e.V.

Goh, G. M. (2010). Space safety standards in Europe. Space Safety Regulations and Standards, 29–48. doi: 10.1016/b978-1-85617-752-8.10003-0

Great Britain. Treasury. Central Computer & Telecommunications Agency (2000). SSADM Foundation. Stationery Office

Highsmith, J. A., 2000, Adaptive software development: a collaborative approach to managing complex systems, Dorset House Publishing Co.

Johansson, B., Austrin, L., Engdahl, G., and Krus, P. (2004). "Tools and Methodology for Collaborative Systems Design Applied on More Electric Aircraft". In: 24th International Congress of the Aeronautical Sciences, 29 August - 3 September 2004, Yokohama, Japan.

Kaisti M, Rantala V, Mujunen T, Hyrynsalmi S, Könnölä K, Mäkilä T, Lehtonen T. Agile methods for embedded systems development – a literature review and a mapping study. In: EURASIP Journal on Embedded Systems 2013; 15: 1–16.

Kroll, P., & Kruchten, P. (2003). Rational Unified Process Made Easy: A Practitioners Guide to the Rup, The. Addison-Wesley Professional.

Kroo, I. (2004). "Distributed Multdisciplinary Design and Collaborative Optimization". In: VKI lecture series on Optimization Methods & Tools for Multicriteria/Multidisciplinary Design, 15–19 November.

Kroo, I., Altusand, S., Braun, R., Gage, P., and Sobieski, I. (1994). "Multidisciplinary Optimization Methods for Aircraft Preliminary Design". In:

5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 7–9 September, Panama City, Florida. AIAA 94-4325.

Larman, C., Basili, V. R., 2003, Iterative and incremental developments. a brief history, Computer (Volume: 36 , Issue: 6 , June 2003), IEEE, pp. 47-56

Lauzier, C.-A. (2014). Requirements management and MOD closure process study on the A330 WV80 project (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-159279

MacConnell, S. (1996). Rapid development: taming wild software schedules. Redmon: Microsoft Press.

McFarlane, N., 2004, Rapid application development with Mozilla, Pearson Education.

Moir, I., & Seabridge, A. (2008). Aircraft Systems mechanical, electrical, and avionics subsystems integration (3rd ed.). New York: Wiley.

Moullin, M. (2002), Delivering Excellence in Health and Social Care, Open University Press,Buckingham

Neely, A.D., Adams, C. and Kennerley, M. (2002), The Performance Prism: The Scorecard for Measuring and Managing Stakeholder Relationships, Financial Times/Prentice Hall, London

Novák, I., 2011, Beginning Visual Studio® LightSwitch Development, Wrox Press.

Palmer, S., & Felsing, J. M. (2002). A practical guide to feature-driven development. Upper Saddle River, NJ: Prentice Hall PTR.

Pardessus, T. (2004). "Concurrent Engineering Development and Practices for Aircraft Design at Airbus". In: Proceedings of the 24th ICAS Conf., Yokohama, Japan.

Parson, S. (2015). A Touch of Class (and Category). FAA Safety Briefing, May/June, 22–25.

Paul, D. and Pratt, D. (2004), History of Flight Vehicle Structures 1903-1990, Journal of Aircraft, Vol. 41, No. 5, September-October.

Phoenix integration, Inc. (July 2010). Phoenix integration, Inc. website. URL: http://www.phoenix-int.com.

Presley, Adrien & Mills, John & H. Liles, Donald. (1998). Presley, A., J. Mills and D. Liles (1995). "Agile Aerospace Manufacturing". Nepcon East 1995, Boston.

Pries, K. H., & Quigley, J. M. (2011). Scrum project management. Boca Raton, FL: CRC Press.

Project Management Institute. (2017a). A guide to the project management body of knowledge: (Pmbok® guide), 6th edition. Newtown Square, PA, USA.

Project Management Institute. (2017b). Agile Practice Guide, 1st edition. Newtown Square, PA, USA.

Ribeiro, C. G. (2017). CAPÍTULO 6 DESENVOLVIMENTO TECNOLÓGICO NACIONAL: O CASO KC-390. In A. T. Rauen (Author), Políticas de inovação pelo lado da demanda no Brasil. Brasília: IPEA.

RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification. Washington, 2011, RTCA Inc.

RTCA DO-254 Design Assurance Guidance for Airborne Electronic Hardware, Washington, 2000, RTCA Inc.

Schwaber, K.; Sutherland, J. (2016) "The Scrum Guide" https://www.scrum.org/resources/scrum-guide

Serrador, P., & Pinto, J. K. (2015). Does Agile work? — A quantitative analysis of Agile project success. International Journal of Project Management, 33(5), 1040–1051. doi: 10.1016/j.ijproman.2015.01.006

Shatil, A.; Hazzan, O.; Dubinsky, Y. Agility in a large-scale system engineering project: a case study of an advanced communication system project". In: Proc. of IEEE International Conference on Software Science, Technology and Engineering, Herzlia (Israel), 2010 :47–54, .DOI: 10.1109/SwSTE.2010.18.

Sobieski, J. (1984). "Multidisciplinary System Optimization by Linear Decomposition". In: Recent Experiences in Multidisciplinary Analysis and Optimization, Symposium held at NASA Langley Research Center, Hampton, Virginia April 24–26. NASA-CP 2327, Part I, pp. 343-366.

Spitz, W.; Golaszewski, R.; Berardino, F.; Johnson, J.; Development Cycle Time Simulation for Civil Aircraft, January 2001 :1-1, NASA/CR-2001-210658.

Takeuchi, H.; Nonaka, I. "The New New Product Development Game." In: Harvard Business Review 64, no. 1 (January–February 1986).

Tschohl, John (2015), " Effective Leadership vs. Management", Agency Sales Vol 45, Issue 2, 42-45, ProQuest

Upadhaya, B., Munir, R., & Blount, Y. (2014). Association between Performance Measurement Systems and Organisational Effectiveness. International Journal of Operations & Production Management, 34(7), 2-2

Wright, T. P. (1945), Aviation's Place in Civilisation, 33rd Wilbur Wright Memorial Lecture, Royal Aeronautical Society Journal, Vol. 49, 1945, p. 299

Wright 1903 Aircraft Engine. (2018, November 20). Retrieved April 23, 2020, from https://wright.nasa.gov/airplane/eng03.html