UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RAPPORT D'ACTIVITÉ DE SYNTHÈSE PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAITRISE EN INFORMATIQUE DE GESTION

PAR
MAC-FRANTZ NOAILLES

AOÛT 2018

REMERCIEMENTS

Cette section est optionnelle.

DÉDICACE

Cette section est optionnelle.
Deuxième ligne de la dédicace.

AVANT-PROPOS

Cette section est optionnelle.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTINGS

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

SWEBOK Guide
TPO Très petite organisation
UQAM Université du Québec à Montréal

RÉSUMÉ

Exemple avec une petite note de bas de page. [1]

1. Ceci est le texte de la petite note.

INTRODUCTION

Le développement de logiciel de qualité est équivalent à la quête du Saint Graal, car on se demande si ce n'est pas un mythe. Or, de nos jours les logiciels prolifèrent dans nos vies quotidiennes. Ils permettent de nous rendre plus efficaces, de communiquer avec nos collègues, nos familles et amis. Ils facilitent notre quotidien. Pensons aux multiples logiciels dans nos appareils portatifs tels nos téléphones intelligents. Pensez à Waze1 [2], P$2 [3] associé à Stationnement de Montréal, mais aussi aux Microsoft [4] et Facebook [5].

Avant d'être les titans d'aujourd'hui Microsoft, Facebook ont vu le jour dans des garages, des dortoirs d'université. Ils étaient tous de très petites organisations à leurs débuts. Il est donc important de soutenir ces types d'organisations. D'un autre côté, comme utilisateur, le public s'attend à ce que les applications offertes par des très petites organisations (après TPO) remplissent ses attentes ou à tout le moins soient à la hauteur des prétentions de leurs développeurs. Par conséquent, les logiciels développés par des TPO se doivent être de qualité. Nous ne pouvons pas prédire les bonnes idées de logiciels et encore moins le succès d'une organisation, mais, il y a-t-il un moyen d'augmenter les chances de succès des très petites organisations. Peut-on faciliter l'atteinte de la qualité ?

Une manière de s'assurer d'avoir un produit de qualité est d'avoir un processus de développement de qualité. Les grandes organisations ont le choix des moyens pour optimiser leurs processus. Elles disposent autant des outils, des ressources financières que du personnel nécessaire pour optimiser leur processus de développement. Les choix des très petites organisations doivent être judicieux, car leurs ressources et le personnel dont ils disposent sont moindres. Ils peuvent difficilement mettre en place les structures ou des processus comparables aux grandes organisations. Par ailleurs, ces structures et ces processus ne leur conviendraient pas. De surcroit, devant la multitude de propositions des outils, méthodes, normes, modèles, méthodologies comment faire un choix éclairé ?

2. www.waze.com
3. www.statdemtl.qc.ca/fr/infos-pratiques/pservicemobile.html
4. www.microsoft.com
5. www.facebook.com

Dans le cadre de cette étude, je mets de l'avant un cadre conceptuel pour permettre à une très petite organisation d'avoir une vision globale de ses efforts d'amélioration de son processus de développement. Je fais et je teste une proposition théorique visant à guider les très petites organisations dans le choix et l'intégration des moyens à mettre en place pour améliorer ce processus.

L'Agilité et la norme ISO/IEC 29110 (après la 29110) sont des moyens à la portée des TPO pour améliorer leur processus de développement. Cependant, les perceptions porteraient à croire qu'ils sont incompatibles. On entend souvent que l'on ne documente pas lorsque l'on fait du développement Agile et la 29110 décrit la documentation minimale afin de fournir une assurance de la qualité d'un logiciel. Toutefois, les principes et les valeurs Agile n'excluent pas la documentation. Plusieurs études démontrent qu'individuellement elles optimisent le processus des organisations. Il serait donc intéressant d'étudier leurs effets combinés. Quel type de relation ont-elles ? Une très petite organisation conforme à la norme aurait-elle avantage à chercher à respecter les principes et valeurs Agile ? Une très petite organisation respectant les principes et valeurs Agile aurait-elle avantage à chercher à se conformer aux exigences de la 29110 ?

CHAPITRE 1.

PROBLÉMATIQUE

Dans cette section, j'expose le sujet de mon étude puis je fais un survol de l'état de la recherche. Le logiciel de qualité est la quête du saint Graal dans le domaine de l'informatique. Premièrement, la pile informatique. Les années ont vu apparaître et disparaitre toutes sortes langages de programmation passant du procédural à l'orienté objet. Plus récemment, les langages de quatrième génération élimineront à ce qu'il paraît les programmeurs, car les usagers seront en mesure d'écrire des programmes en se basant sur leurs langages d'affaires. Que dire des promesses de l'intelligence artificielle qui apprendra et se programmera seule ? Deuxièmement, les méthodes et les méthodologies. Les années ont vu apparaître et disparaitre toutes sortes de méthodes et de méthodologies pour améliorer le processus en amont au moment de la réalisation des activités. Les plus récentes méthodes, pour ne nommer que cela, sont Scrum et XP, AUP. Les méthodologies, elles, sont passées de cascade à itérative. Troisièmement, les normes, les modèles et les cadres. Les années ont également vu naître, évoluer et des fois disparaître une multitude de normes, de modèles et de cadres qui permettent de s'assurer que les activités importantes ont été accomplies correctement. Les plus connus et les plus récentes étant, Cobit, ITIL, CMMI et la 29110. Depuis des décennies, un débat fait rage entre ce qu'on appelle les approches légères « light weight » et celles dites disciplinées. Les approches légères relevant des méthodologies et des méthodes et celles disciplinées relevant des modèles, normes, etc. On en est au point où où être tenant de l'un signifie être contre l'autre. Toutefois, malgré des décennies de recherche à la quête de la solution miracle force est de constater qu'aucun des outils, normes, méthodes ou méthodologies, etc. n'arrive à prétendre avoir trouvé le « silver bullet » (Brooks, 1986). Il serait faux cependant de dire que les recherches sont vaines et encore moins que les solutions mises de l'avant n'aient rien améliorées ou encore n'apporteront plus rien. Le résultat est que la recherche dans le domaine de l'optimisation des activités est marquée par ce conflit idéologique et la recherche de la solution miracle. Elle est réalisée de manière fragmentée, spécialisée et en vase clos dans laquelle est pratiquée. Les formes recherches dans le domaine sont

pour l'essentiel de la recherche appliquée qui « consiste à trouver des solutions à des problèmes pratiques et que la connaissance peut être immédiatement axée sur l'action ou la prise de décision. » (Fortin, Gagnon, Fournier et Lauzier, 2016, p. 19). Mon avis est que la plus-value à attendre de ce type de recherche est moindre que celle que l'on tirera de leur agencement de manière judicieuse. La réalité est que les circonstances dans lesquelles le logiciel est développé, les personnes qui en font le développement et le type de logiciel développé présentent une infinité de possibilités. Est-il logique de penser qu'une solution passe partout existe ? D'autre part, après toutes ces années n'a-t-il pas lieu de remettre en question, de mettre en doute nos dogmes ? Un proverbe bien connu dont la provenance est disputée dit ceci : « La folie c'est de faire quelque chose de manière identique et d'espérer des résultats différents » (Traduction libre). L'étude que je propose rejoint plutôt la recherche de type fondamental. Elle vise à permettre aux très petites organisations de prendre du recul par rapport à leurs activités et à les aider dans le choix des améliorations qu'ils y apportent. Suite à un raisonnement déductif, j'en viens à affirmer que pour un niveau de qualité un niveau de qualité donnée du processus de développement tel que défini par le profile basique de la 29110 les efforts d'ingénierie logicielle appliqués aux trois groupes d'activités sont complémentaires. C'est-à-dire que les efforts d'optimisation dans un groupe d'activités permettent de réduire les efforts nécessaires pour optimiser les activités d'un autre groupe.

FIGURE 1.1. – Proposition théorique simplifiée

Ma proposition théorique a été déduite d'un modèle conceptuel basé sur les écrits du SWEBOK et se base sur le postulat qu'un processus de développement logiciel de qualité produit un logiciel de qualité. Dans mon étude, je cherche à explorer cette relation par la vérification d'une hypothèse. Mon hypothèse est que dans le contexte d'une très petite organisation l'implémentation des valeurs et des principes de l'Agilité et des exigences de la 29110 est complémentaire. L'Agilité étant orientée principalement vers les activités primaires et la norme ISO/IEC 29110 l'étant plus vers les activités de support. J'explore donc l'impact de la mise en place de l'un sur l'autre. Si les résultats de mon étude supportent l'hypothèse, cela appuiera la probabilité de la validité de la relation et celle de la théorie. Toutefois, d'autres validations de l'hypothèse et de la théorie s'imposeront.

CHAPITRE 2.

CONCLUSION

Petit texte de la conclusion.

ANNEXE A.

A.1. État de la recherche

Comme je l'ai dit auparavant, il existe peu de recherches de type fondamental sur l'utilisation concertée d'outils, méthodes, normes ou méthodologie et encore moins qui font le pont entre les approches légères et disciplinées. Il existe toutefois certains auteurs qui ont examiné la compatibilité entre l'Agilité certains outils, méthodes, normes ou méthodologie ou vice-versa.

Du côté de l'Agilité dans les études concertées, les auteurs ont surtout cherché à démontrer que l'Agilité était applicable dans le cadre de grande organisation. Ainsi donc, une équipe du Software Engineering Institute (SEI)5 a examiné la compatibilité du modèle CMMI et de l'Agilité (Glazer, Dalton, Anderson, Konrad et Shrum, 2008). Une des conclusions importantes est que si « les professionnels autant du CMMI que de l'Agilité apprenaient et comprenaient leurs différences et explorait les moyens de prendre avantage les uns des autres, nous trouvons [ils trouveraient] des manières de combinées leurs idées pour l'optimisation des processus ils amèneraient ces derniers à des niveaux sans précédent ». Cette équipe a réalisé une comparaison du CMMI et de l'Agilité sur les treize (13) dimensions suivantes : le focus sur l'unité organisationnelle, la gestion, la confiance, la planification, le marché et les attentes, le design et l'architecture, l'apprentissage, la durée de vie de l'équipe, la mesure, le développement du personnel, l'horizon de validation, la gestion du risque, le coût des défauts. Trudel a présenté son analyse de la compatibilité entre SCRUM et CMMI selon les domaines du CMMI soient la gestion de projet, l'ingénierie, la gestion du processus et le support (Trudel, 2006). Al-Elaimat et Al-Ghuwairi, eux, ont cherché à lier les activités spécifiées par la méthode XP avec les exigences de la norme ISO/IEC 12207 et du CMMI. (Al-Elaimat et Al-Ghuwairi, 2015) Ces écrits du fait des sujets comparés ne correspondent pas au contexte de mon étude, mais plutôt celle d'une grande entreprise. Par ailleurs, l'Agilité seule jusqu'en 2008 faisait l'objet de peu de recherche sur les 1996 publications recensé par Dybå et Dingsøyr (2008), cinq (5) était de qualité et portait sur l'Agilité au sens large. Malgré

le recensement plus récent de Dingsøyr, Nerur, Balijepally et Moe (2012), on n'est pas en mesure de statuer si le ratio d'étude sur l'Agilité s'est amélioré.

Du côté de la norme ISO/IEC 29110. Les auteurs ont examiné la compatibilité avec les méthodes Agiles ou encore ont cherché à ajuster les méthodes Agile. Galvan, Mora, O'Connor, Acosta et Alvarez ont analysé la conformité des activités de gestion de projet des méthodes Agile, UPEDU et SCRUM, à la 29110.(Galvan, Mora, O'Connor, Acosta et Alvarez, 2015). Bien que le contexte de leur article ressemble à celui que je vise dans le cadre de mon étude, ces auteurs ont analysé des méthodes Agile plutôt que l'Agilité sur la base de ces principes et valeurs. Par ailleurs, la recherche sur la norme ISO/IEC 29110 seule porte soit sur de nouvelles contributions, et ce souvent dans le cadre du développement de processus d'évaluation et des modèles ou méthodes d'amélioration des processus selon le plus récent recensement de Moreno-Campos, Sanchez-Gordón, Colomo-Palacios et de Amescua Seco (2014).

López-Lira et Hinojo, eux, proposent un modèle conceptuel bidimensionnel pour comparer la couverture des types d'activités du processus de développement et des niveaux de complexité de ce qu'ils nomment des SoP « Set of practices pratiques implémentées (Traduction libre) ». Les SoP qu'ils analysent sont ceux du CMMI, PSP, TSP, SCRUM, XP, AUP, ISO/IEC 12207, ISO/IEC 29110, MoProSoft, PMBOK, Clean Room et RUP. (López-Lira Hinojo, 2014) Ces auteurs proposent un modèle conceptuel dans lequel ils placent différentes méthodes, modèles et normes avec l'objectif de faire des agencements complémentaires. Dans le cadre de mon étude, le modèle conceptuel est basé sur le SWEBOK v3.0. Je n'ai pas su trouver d'autres travaux présentant un modèle conceptuel combinant l'Agilité, une méthode Agile et la norme ISO/IEC 29110.

A.2. : Définitions

Le logiciel Je m'attarde à la définition de logiciel pour deux raisons. D'une part, pour éviter la perception voulant que le terme se rapporte uniquement au code informatique alors qu'il a un sens plus large. D'autre part, la 29110, dans sa version la plus récente, porte sur le logiciel et les systèmes. Je définis donc brièvement ces deux concepts, car par la suite je délaisserai celui de systèmes pour me concentrer sur le logiciel mon étude ne portant que sur les exigences concernant le logiciel. Dans la suite de ce document, les références à la norme ISO/IEC 29110 (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.) indiquent les exigences logicielles de celle-ci.

Un système est une « collection organisée de composants visant à accomplir une fonction ou un ensemble spécifique de fonctions » (April et Laporte, 2011). La norme ISO/IEC/IEEE 15288 : 2015 va dans le même sens et précise qu'un système peut être considéré comme un produit ou par rapport au service qu'il rend. Habituellement, un nom y est associé afin de clarifier son sens, ex. système d'avion. Par ailleurs, le mot système peut être remplacé par un synonyme selon le contexte, ex. avion, mais cela cache les différentes fonctions qui sont accomplies.

La définition du logiciel, elle, est d'intérêt pour l'étude. Plusieurs définitions du logiciel existent. Laporte et April définissent le logiciel comme « un ensemble de programmes, de procédures et éventuellement, de documentation associées et de données qui concernent le fonctionnement d'un système informatique » (April et Laporte, 2011). La norme ISO 24765 va dans le même sens, mais d'une part ajoute des précisions sur les termes utilisés par April et Laporte et d'autre part, ajoute règles, matériaux connexes et « firmware ». La norme ISO/IEC 24765 décrit un logiciel comme un ensemble de programmes, de données, processus, règles, documentation, matériaux connexes et « firmware » micrologiciel. • L'ensemble de programmes étant le code source qui a fait l'objet de spécification, de conception, d'inspection et d'essais unitaires, système et d'acceptation avec la clientèle. • Les données étant celles inventoriées, modélisées, normalisées et créées pour effectuer des scénarios d'essais lors de sa création ou d'une modification. • Les processus étant ceux d'affaires des utilisateurs et les processus qui ont été décrits (avant l'automatisation et après), étudiés et optimisés. Les règles étant les celles d'affaires décrites, validées, implantées et testées. • La documentation étant toute celle utile pour les utilisateurs, développeurs et mainteneurs de logiciel. Elle permet aux différents intervenants d'une équipe de mieux communiquer, réviser et tester le logiciel. Elle est définie et réalisée à toutes les étapes cycle de vie d'un logiciel. • Les matériaux connexes sont les plans et outils de formation, les plans de transition vers la maintenance, les ententes de services et les environnements de production. • Le « firmware » étant la combinaison d'un dispositif matériel et d'instructions d'un ordinateur ou de données informatiques qui résident, en mode lecture seulement, sur un périphérique matériel.(Séguin, 2016, n. Acétat cours 1)

Ce qu'il est important de retenir est que lorsque je parle de logiciel cela comprend plus que le juste code source.

Le cycle de vie du logiciel cycle vie du produit et processus de développement

Mon étude porte sur les activités du processus développement logiciel. Il est important

de bien situer ce processus critique dans le cycle de vie du logiciel afin de comprendre la portée des activités qui la composent. Certaines de ces activités pertinentes pour l'étude seront présentées plus loin.

Les trois concepts, processus de développement « software developpment life cylce » SDLC, cycle de vie du produit « Software product life cycle » SPLC et cycle de vie du logiciel « software life cycle », se confondent souvent dans la littérature surtout en anglais. La raison est que souvent la même terminologie « software life cycle » est souvent utilisée pour les trois.

D'abord le cycle de vie du logiciel ou « Software life cycle ». Le cycle de vie est l'« évolution d'un système, produit, service, projet ou autre entité d'origine humaine de la conception à la retraite (April et Laporte, 2011) ». Le cycle de vie se subdivise en six processus, l'acquisition, la fourniture, le développement, l'opération, la maintenance et la destruction selon ISO (traduction libre) (International Organization for Standardization (ISO), 2008). Petite parenthèse ici pour nuancer le terme « destruction » et marquer la différence entre la maintenance et l'évolution. De nos jours, surtout avec le développement orienté objet ou orienté service, le logiciel est rarement mis au rancart. Le code est souvent réutilisé en tout ou en partie. Donc, il n'y a pas de mise aux rebus ou de disparition comme le terme pourrait laisser à croire. À sa place le SWEBOK utilise la mise à la retraite, un terme qui laisse entrevoir que l'utilisation active du code arrête, mais le code demeure disponible au besoin. Il indique que le cycle de vie comprend un plus du processus de développement ceux de : déploiement, maintenance, support, évolution, mise à la retraite et tous les autres processus entre le lancement et la mise à la retraite. (Bourque et Fairley, 2014, p. 151). D'autre part, pour différencier le cycle de maintenance de celui d'évolution, on se fie à l'ampleur et le type de changement apporté. Pour une discussion plus exhaustive, voir le livre de Tripathy et Naik (2014).

Le processus de développement ou « Software development life cycle » est donc une phase du cycle de vie du logiciel. Ainsi au cours du cycle de vie d'un logiciel, il est possible que ce dernier passe au travers de multiples phases de développement. J'illustre mon propos à la Figure 1.2

Enfin, il ne faut pas confondre le cycle de vie que je viens de présenter qui englobe tous les cas de figure et le cas spécifique du cycle de vie d'un produit ou « Software product life cycle ». Ainsi, un PGI n'aura pas le même cycle de vie qu'une application embarquée sur la puce qui régit le jeu de lumière d'une botte d'enfant. Dans ce dernier cas, il n'y

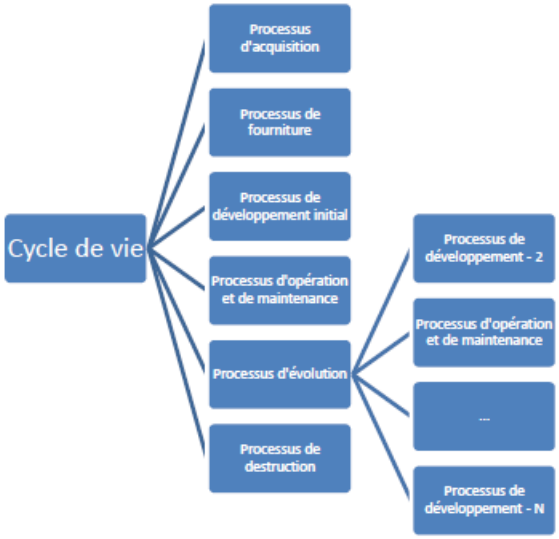

FIGURE A.1. – Cycle de vie vs cycle de développement

aura fort probablement pas d'opération, de maintenance et encore moins de l'évolution. Une discussion des différents modèles de cycle de vie dépasse le cadre de cette étude.

Il est donc important de retenir deux (2) choses en ce qui a trait au processus de développement. Premièrement, il s'agit d'un processus récurrent dans le cadre du cycle de vie du logiciel. L'efficacité de ce processus a un impact majeur autant sur le logiciel, sur l'organisation qui en a la charge et sur l'entreprise dans son ensemble. Aussi, elle a un impact sur les coûts des processus d'opération, de maintenance et d'évolution. Deuxièmement, la séquence des processus des cycles de vie du logiciel est variable c'est-à-dire qu'un logiciel ne passe pas nécessairement par tous les processus. Un logiciel acheté par une organisation, par exemple, n'aura pas de cycle de développement initial pour cette organisation, mais fera l'objet du processus d'opération et de maintenance et peut être l'objet d'évolutions par la suite.

Le processus de développement logiciel Le SWEBOK indique que ce processus regroupe six (6) catégories d'activités, les primaires, ceux de support, les organisationnels, ceux qui sont transversaux aux projets en cours, ceux de gestion de projet et, enfin, ceux concernant des besoins particuliers.

Les processus primaires comprennent ceux du processus de développement, d'opération et de maintenance du logiciel, les processus de support sont appliqués de manière intermittente ou continuelle durant le cycle de vie du produit en support aux processus

primaires, ils sont la gestion de la configuration, l'assurance qualité ainsi que celui de vérification et validation. Les processus organisationnels apportent un support à l'ingénierie logicielle, ils sont la formation, l'analyse de la mesure, la gestion de l'infrastructure, la gestion de portefeuille et de la réutilisation, la gestion du changement organisationnel et la gestion des modèles de cycle de vie. Les processus transversaux telles la réingénierie, les gammes de produits et l'ingénierie de domaine s'appliquent à plus d'un projet de l'organisation… Les activités de la catégorie des besoins particuliers telles celles qui ont trait aux caractéristiques de qualité regroupent, par exemple, les besoins de sécurité durant le processus de développement qui peuvent exiger la réalisation tâches pour protéger l'environnement de développement et réduire les risques d'actes malveillants. D'autres tâches peuvent être réalisées afin d'offrir une assurance sur l'intégrité du logiciel. (Traduction libre) (Bourque et Fairley, 2014, p. 152) J'illustre la portée des catégories d'activités au Tableau 1.1.

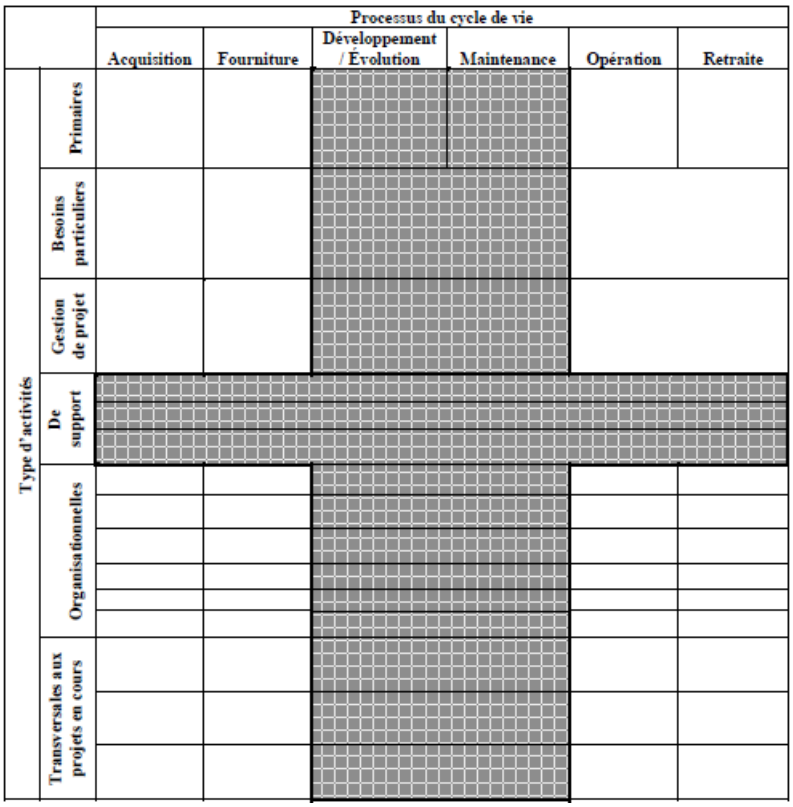

FIGURE A.2. – Portée des catégories d'activités dans le cycle de vie

Enfin, il est important de noter deux (2) choses. D'une part, Le SWEBOK apporte une précision aux activités de support. Alors que les autres activités se déroulent dans

le cadre de deux (2) processus soient le développement et la maintenance celles de support, elles, se déroulent à l'intérieur du cycle de vie du logiciel donc au-delà des deux (2) processus. D'autre part, il est important de noter que le SWEBOK traite de façon distincte les activités liées à la catégorie gestion de projet. Je reviens plus loin dans cette sous-section sur les activités de cette catégorie.

Les activités primaires du processus de développement logiciel La définition du SWEBOK n'indique pas de façon explicite quelles sont les activités primaires du processus de développement. La 29110, elle, indique que six (6) activités en font partie :

SI.1 Initiation de la mise en oeuvre du logiciel. SI.2 Analyse des exigences du logiciel SI.3 Architecture du logiciel et conception détaillée du logiciel SI.4 Construction du logiciel SI.5 Intégration et tests du logiciel SI.6 Livraison du produit (International Organization for Standardization (ISO), 2014, p. 24)

Du côté de l'Agilité, Boisvert et Trudel indiquent qu'il faut que les disciplines du développement logiciel soient exécutées en parallèle plutôt qu'en séquence. Chaque itération comporte donc un certain lot d'activités de planification, d'analyse, de conception, d'écriture de code, de tests et de livraison. (Boisvert et Trudel, 2011)

Dans l'ensemble, les énumérations d'activités sont semblables en considérant que l'activité de planification et d'initialisation de la mise en oeuvre revoit aux mêmes types d'activité. Il reste que la 29110 fait une différence entre les activités d'intégration et les tests. Cette différence renvoie aux tests unitaires et aux tests d'intégration qui peuvent être compris dans les activités de test qu'indiquent Boisvert et Trudel.

Les activités de gestion de projet La norme fait référence à la gestion de projet en ces termes : « Le but du processus de gestion de projet (Project Management [PM]) consiste à établir et à réaliser, systématiquement, les Tâches inhérentes au projet de déploiement du logiciel, lesquelles permettent d'assurer la conformité avec les Objectifs du projet en matière de qualité, de durée et de budget. » (International Organization for Standardization (ISO), 2014, p. 4)

Boisvert et Trudel indiquent que dans le cadre de l'Agilité, la gestion de projet « C'est une façon de s'assurer que toutes les activités nécessaires pour atteindre les objectifs du projet tiendront à l'intérieur des contraintes de temps et de budget. » (Boisvert et Trudel, 2011, l. 1942) Encore là les définitions se ressemblent sauf que la 29110 ajoute un objectif de qualité. Cette différence s'explique par l'approche de la 29110 par rapport à l'Agilité. Je discute de cette différence dans la sous-sous-section sur le modèle conceptuel

qui suit. Pour l'instant, je dirai tout simplement que l'Agilité s'efforce d'incorporer la qualité, en amont, durant la réalisation des activités, alors que la 29110 s'assure en aval, une fois l'activité exécutée, que la qualité est présente. Enfin, le SWEBOK indique que les activités liées à cette catégorie sont « la planification et l'estimation, la gestion des ressources, la mesure et le contrôle, le leadership, la gestion de risque, la gestion de parties prenantes et la coordination des activités primaires, de support, organisationnelle et transversal aux projets durant les processus de développement et de maintenance (Traduction libre) ». (Bourque et Fairley, 2014, p. 152)

Basé sur la description des catégories d'activités que nous venons de voir, je peux compléter le Tableau 1.2 décrivant les différentes activités associées au processus de développement logiciel.

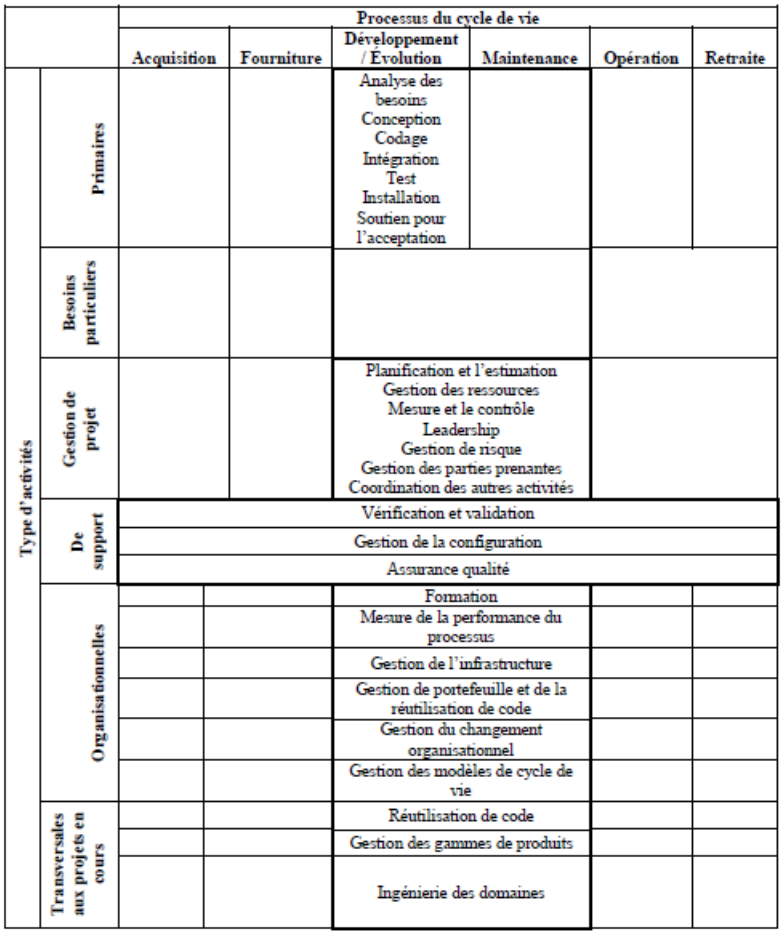

FIGURE A.3. – Activités du processus de développement logiciel

L'ingénierie logicielle Il est important de distinguer le processus développement du logiciel que nous venons de voir de son ingénierie. Le premier porte sur la réalisation alors que le second porte sur la manière de réaliser le logiciel. Comme je l'ai déjà présenté, le processus de développement logiciel est un « Processus suivant lequel les besoins de l'utilisateur sont traduits sous forme de configuration logicielle, celle-ci est incorporée dans la conception et la conception est mise en oeuvre sous forme de code ; celui-ci, enfin, est testé, documenté et certifié pour l'exploitation (Gouvernement du Canada, 2016a) ». Tandis que l'ingénierie logicielle est l'« Application systématique des connaissances, des méthodes et des acquis scientifiques et techniques pour la conception, le développement, le test et la documentation de logiciel afin d'en optimiser la production, le support et la qualité (Gouvernement du Canada, 2016b) ».

Mon étude s'intéresse à l'ingénierie logicielle telle qu'appliquée dans le cadre du processus de développement, car dans ces deux définitions il est important de souligner deux différences fondamentales. D´abord, l'ingénierie logicielle ne se limite pas au processus de développement du cycle de vie du logiciel. Elle vise l'optimisation de la production donc le développement, mais également l'optimisation du support et de la qualité. Le SWEBOK va dans le même sens lorsqu'il dit que le processus d'ingénierie logicielle : se préoccupent les activités accomplies par les informaticiens pour développer, maintenir et opérer le logiciel, tel que la cueillette des exigences, le design, la construction, les tests, la gestion de la configuration, ainsi que des autres processus…pour nombre de raison : faciliter la compréhension, la communication et la coordination ; aider à la gestion des projets ; pour mesurer et améliorer la qualité du logiciel de manière efficace ; pour supporter l'amélioration des processus ; et comme premier pas à l'automatisation des activités (Traduction libre) (Bourque et Fairley, 2014, p. 148)

L'Agilité par le biais des méthodes qui en relèvent, KANBAN, SCRUM, AUP, par exemple, de même que la 29110 sont de l'ingénierie logicielle appliquée au processus de développement. Ils sont appliqués aux activités des trois groupes du processus de développement. Je rentrerai dans les détails dans le cadre de la présentation de l'Agilité et de la 29110 en commençant par la 29110.

L'Agilité Plusieurs méthodes et cadres sont dits Agiles entre autres, SCRUM, KANBAN, AgileUP, XP, SAFe, DAD et LeSS. Ces méthodes et cadres appliquent les principes Agile à différentes activités et à différentes tâches au sein de ces activités du processus de développement. Les quatre (4) valeurs de l'Agilité sont (Beck et al., 2001a) : ~~~ Les individus et les interactions davantage que les processus et les outils. Des logiciels opérationnels plus qu'une documentation exhaustive. La collaboration avec le client plus que

la négociation de contrat. L'adaptation au changement plus que le suivi d'un plan. ~~~ Les douze (12) principes sont (Beck et al., 2001b) : ~~~ Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre lesobjectifs fixés. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face. Un logiciel opérationnel est la principale mesure d'avancement. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle. Les meilleures architectures, spécifications et conceptions émergent d'équipes autoorganisées. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence. ~~~

La norme ISO/CEI 29110 Généralités Les guides, développés par le groupe de travail de l'ISO, sont basés sur des sous-ensembles d'éléments des normes appropriées, appelés « profils ». Le but d'un profil est de définir un sous-ensemble de normes ISO/IEC pertinentes pour le contexte des TPO, par exemple, les processus de l'ISO/IEC/IEEE 15288 ou de l'ISO/IEC/IEEE 12207 et des produits de la norme ISO/IEC/IEEE 15289.(Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

La figure ci-dessous illustre le positionnement de l'ISO/IEC 29110 dans le spectre des approches de développement. (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

Structure des documents de la norme La norme se compose de cinq (5) types de documents, l'aperçu général, le cadre général et taxinomie, les spécifications de profils, le guide d'évaluation et les guides d'ingénierie et de gestion (voir la Figure 2.8).

Ces documents encadrent les TPO faisant pour l'instant partie de quatre (4) audiences ou groupes et ayant un profil donné tel que décrit au Tableau 2.1

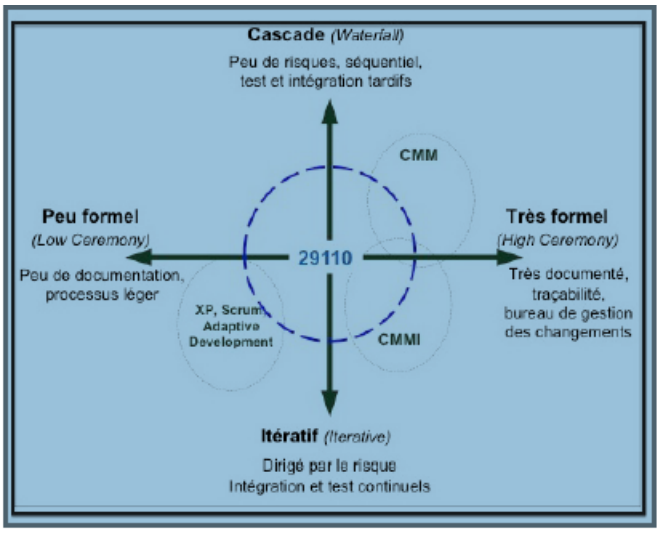

FIGURE A.4. – Les approches de développement et l'ISO/IEC 29110 (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

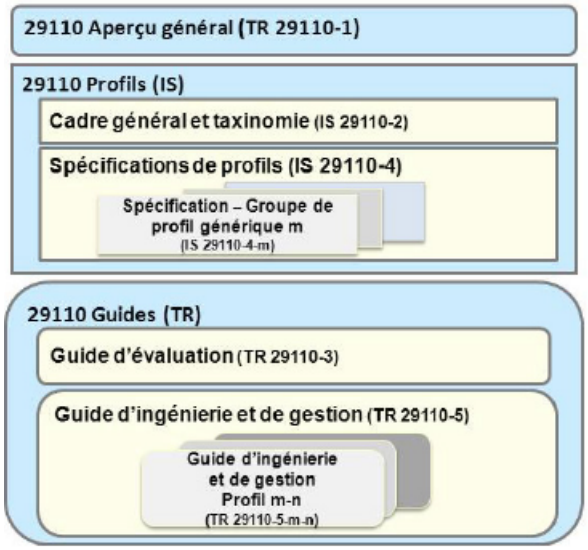

FIGURE A.5. – Série ISO/CEI 29110 (Organisation internationale de Normalisation, 2012)

Groupe de Profil	Profil	Partie de spécification réf.	Partie du guide part réf.
29110.1 – Ingénierie logiciel — Générique	29110.1.1 - Entrée	ISO 29110-4-1, Article 7	ISO/TR 29110-5-1-1
	29110.1.2 - Basique	ISO 29110-4-1, Article 8	ISO/TR 29110-5-1-2
	29110.1.3 - Intermédiaire	ISO 29110-4-1, Article 9	ISO/TR 29110-5-1-3
	29110.1.4 - Avancé	ISO 29110-4-1, Article 10	ISO/TR 29110-5-1-4
29110.2 Organisationnel	29110.2 - Organisationnel	ISO 29110-4-2, Article 7	ISO/TR 29110-5-2
29110.3 – Livraison de service	29110.3.1 - Gouvernance	ISO 29110-4-3, Article 7	ISO/TR 29110-5-3
	29110.3.2 – De support	ISO 29110-4-3, Article 8	ISO/TR 29110-5-3
	29110.3.3 - Continuel	ISO 29110-4-3, Article 9	ISO/TR 29110-5-3
29110.6 – Ingénierie des systèmes – Générique	29110.6.1 – Entré	ISO 29110-4-6, Article 7	ISO/TR 29110-5-6-1
	29110.6.2 - Basique	ISO 29110-4-6, Article 8	ISO/TR 29110-5-6-2
	29110.6.3 - Intermédiaire	ISO 29110-4-6, Article 9	ISO/TR 29110-5-6-3
	29110.6.4 - Avancé	*ISO 29110-4-6, Article 10*	*ISO/TR 29110-5-6-4*
NOTE Profil et documents en italique ont été identifiés, mais le projet ne l'a pas été au moment de la publication.			

FIGURE A.6. – Taxonomie de profile de TPO traduit de (International Organization for Standardization (ISO), 2016a)

La norme d'ingénierie logicielle pour TPO Le groupe d'ingénierie logiciel qui nous intéresse dans le cadre de cette étude présente quatre (4) profils, d'entrée, basique, intermédiaire et avancée.

Dans le cadre de cette étude, le profil d'intérêt est le profil basique. Il se compose de deux processus, de gestion de projet et de mise en oeuvre du logiciel, décrite à la Figure 2.9 Différentes trousses de déploiement ont été développées spécifiquement pour les très petits organismes afin de les aider à mettre en oeuvre le profil basique.

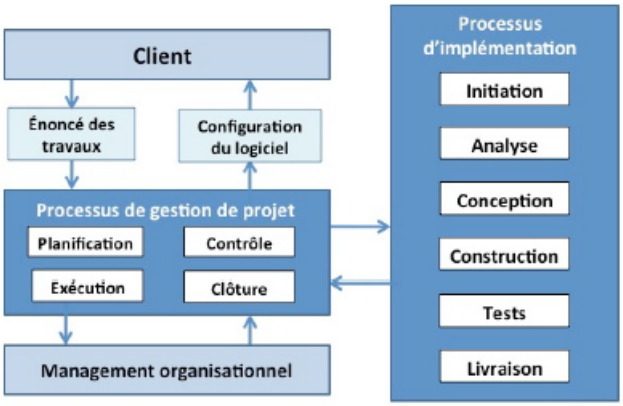

FIGURE A.7. – Les 2 processus et les activités du profil basique de l'ISO/IEC 29110 (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

Les caractéristiques des TPO ciblés par ce profil Le profil basique décrit le développe-

ment de logiciels d'une seule application par une équipe de projet unique sans risque ou des facteurs conjoncturels spéciaux. Le projet peut être de remplir un contrat externe ou interne. Le contrat interne n'a pas à être explicite entre l'équipe du projet et son client. Pour utiliser le Guide (c.-à-d. la partie 5-1-2), un TPO doit satisfaire aux conditions d'entrée suivantes : ~~~ Un énoncé des travaux (Statement of work) est documenté, Une étude de faisabilité du projet a été réalisée avant son lancement, L'équipe de projet, y compris le chef de projet, a été assignée et formée, et Les biens, les services et l'infrastructure pour démarrer le projet sont disponibles (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.) ~~~ Le processus de gestion de projet Le but du processus de gestion de projet est d'établir et de mener à bien de façon systématique, les tâches du projet de mise en oeuvre du logiciel, ce qui permettra de répondre aux objectifs du projet en ce qui concerne la qualité, le calendrier et le coût. (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.) Le processus de gestion de projet comporte les activités suivantes illustrées à la Figure 2.10 : • PM.1 Planification du projet • PM.2 Exécution du plan du projet • PM.3 Évaluation et contrôle du projet • PM.4 Clôture du projet (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

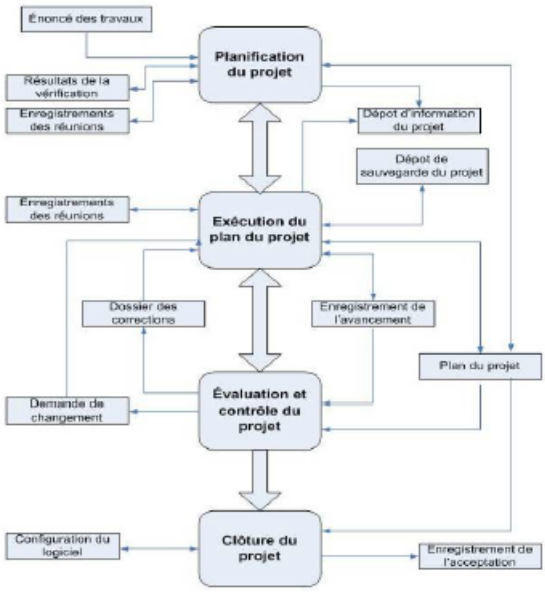

FIGURE A.8. – Processus de gestion de projet du profil basique (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

Le processus de mise en oeuvre du logiciel Le but du processus de mise en oeuvre de logiciels est la performance systématique des activités d'analyse, de conception, de construction, d'intégration et de tests pour les produits logiciels nouveaux ou modifiés

selon les exigences spécifiées. (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.) Le processus de mise en oeuvre comporte les activités suivantes (voir Figure 2.11) : • SI.1 Initiation de la mise en oeuvre du logiciel • SI.2 Analyse des exigences du logiciel • SI.3 Architecture et conception détaillée du logiciel • SI.4 Construction du logiciel • SI.5 Intégration et tests du logiciel • SI.6 Livraison du produit

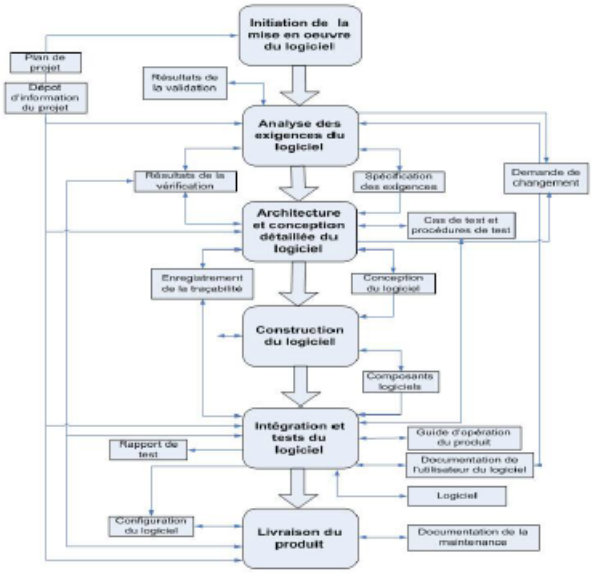

FIGURE A.9. – Processus de mise en oeuvre du logiciel du profil basique (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

Les trousses de déploiement Le but de ces trousses est de fournir des guides détaillés (p.ex. des tâches, des étapes, des rôles et des produits, des gabarits, des listes de vérification) aux TPO. Les TPO peuvent ainsi sélectionner des parties du référentiel ISO 29110 sans avoir à toutes les mettre en oeuvre en même temps (voir la). Les trousses disponibles gratuitement sur internet pour le profil basique (Lebel et Laporte, 2016) : • Construction et tests unitaires : Cette trousse concerne les activités de développement de code et de tests unitaires du logiciel. • Gestion du projet : Cette trousse présente les activités liées à la gestion de projet, soit la planification, l'exécution, l'évaluation/-contrôle et la clôture du projet. • Analyse des exigences : Cette trousse de déploiement s'intéresse à l'activité d'Analyse des exigences du logiciel qui consiste à analyser les exigences établies en accord avec le client et à valider la portée du projet. • Vérification et validation : Cette trousse présente les activités qui permettent de vérifier la conformité du logiciel livré et de valider que le produit satisfasse les besoins du client. • Architecture et conception détaillée : L'activité d'architecture et de conception détaillée consiste

à identifier et à organiser les différents composants d'un logiciel en une structure cohérente. • Contrôle des versions : Cette trousse explique les étapes pour gérer les versions du code source. • Intégration et tests : Cette trousse décrit l'activité d'intégration et de tests du logiciel. Cette activité permet de vérifier le bon fonctionnement des produits logiciels développés et d'éviter les dommages que peuvent causer les anomalies non détectées. • Livraison du produit : Cette trousse définit les instructions de livraison avec le client et formalise la fin du projet. • Auto-évaluation : Cette trousse permet de comparer les intrants et extrants de chacune des tâches spécifiques des activités l'ISO 29110. Elle offre un tableau de bord qui présente des graphiques d'évaluation des processus de gestion de projet et d'implémentation logicielle.

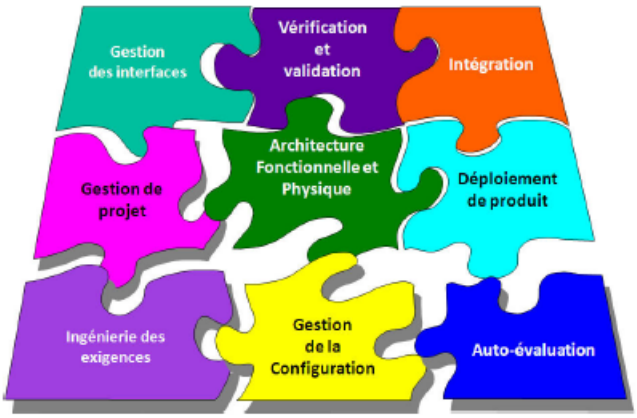

FIGURE A.10. – Trousses de déploiement proposées pour le profil basique en ingénierie des systèmes (Groupe de travail 24 de ISO/IEC JTC1/SC7, s.d.)

A.3. : Proposition théorique générale

Jusque-là la discussion a permis de situer les différents concepts de l'étude et leurs relations dont la relation entre le processus et les activités de développement ainsi que l'ingénierie logicielle. Sur la base des descriptions précédentes, je peux donc conclure que le processus de développement logiciel tel que décrit par le SWEBOK est une combinaison des processus primaires spécifiques au processus, des activités de gestion de projet, organisationnelles et transversales nécessaires dans le cadre du développement/évolution ou de la maintenance et enfin des activités de support dans le cadre du cycle de vie du logiciel.

Ce modèle est utile, surtout sa représentation sous forme de tableau détaillant les acti-

FIGURE A.11. – Modèle conceptuel général basé sur le SWEBOK

vités, pour faire l'inventaire des activités du processus de développement encadrées le processus d'ingénierie logicielle soit les méthodes, modèles ou des normes.

Catégories	Activités du processus de développement	
Primaires	Analyse des besoins	(1)SCRUM, XP
	Conception	(1)XP
	Codage	(1)XP
	Intégration	(1)SCRUM, XP
	Test	(2)29110
	Installation	
	Soutien pour l'acceptation	
Besoins particuliers		
Gestion de projet	Planification et l'estimation	SCRUM (-estimation)
	Gestion des ressources	SCRUM
	Mesure et le contrôle	
	Leadership	SCRUM
	Gestion de risque	SCRUM, (3)29110
	Gestion des parties prenantes	SCRUM
	Coordination des autres activités	
De support	Vérification et validation	(4)29110
	Gestion de la configuration	
	Assurance qualité	(5)29110
Organisationnel	Formation	
	Mesure de la performance du processus	
	Gestion de l'infrastructure	
	Gestion de portefeuille et de la réutilisation	
	Gestion du changement organisationnel	
	Gestion des modèles de cycle de vie	
Transversaux	Gestion de la réutilisation	
	Gestion des gammes de produits	LeSS, DAD, SAFe
	Ingénierie des domaines	
Note 1 : Figure 2.4 | Note 2 : Claude Yvon Laporte et April (2011, p. 62) | Note 3 : Claude Yvon Laporte et April (Claude Yvon Laporte et April, 2011) | Note 4 : Voir April et Laporte (2011, p. 298) |Note 4 : April et Laporte (2011, p. 172)		

FIGURE A.12. – Couverture des activités du processus de développement général

Toutefois, il ne donne aucune idée des relations qui peut exister entre les concepts. Sur ce dernier point, nous avons le postulat sur la relation entre l'ingénierie logicielle, le processus et les activités de développement qui éclaire. Pourquoi l'ingénierie logicielle est-elle importante ? En d'autres termes pourquoi l'Agilité et la 29110 sont-elles importantes ? La réponse est que le développement, la maintenance et l'évolution d'un logiciel sont des processus complexes et des processus d'ingénierie logicielle les optimisent et augmentent les chances d'avoir un produit de qualité. L'ingénierie logicielle, dont l'Agilité et la 29110 permettent d'avoir un effet multiplicateur sur la qualité du processus de développement

en optimisant ses activités.

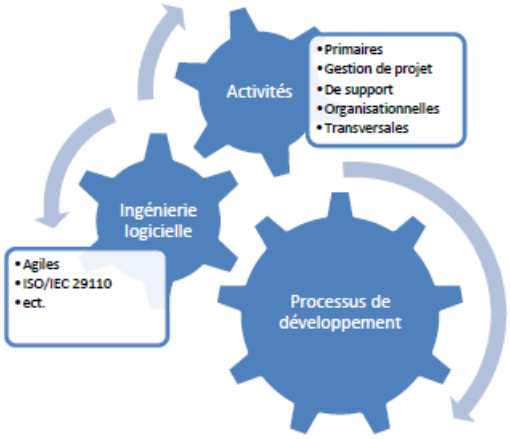

FIGURE A.13. – Effet multiplicateur de l'ingénierie logicielle

Autrement exprimé, on peut affirmer que les efforts d'ingénierie appliqués aux différentes activités du processus de développement résultent au niveau de qualité du processus. Cette relation est celle que nous avons vue au début de la problématique est reprise à nouveau à la ci-après.

FIGURE A.14. – Proposition théorique générale

Cette relation est trop large et vague pour être validée et je suis rendu au bout du processus de raisonnement déductif. Les relations décrites par des macrothéories (Fortin et al., 2016, p. 102) sont trop larges et complexes pour être étudiées. Selon le processus de raisonnement déductif, on part d'une explication générale (généralisation empirique) pour se diriger vers un évènement particulier. Pour élaborer une théorie selon cette approche, on formule une question de recherche en s'inspirant d'une théorie de portée générale ou d'une proposition hypothétique, puis on émet des hypothèses concernant le phénomène précis, et ces dernières sont soumises à vérification empirique.(Fortin et al., 2016, p. 106) Dans la sous-sous-section suivante, je propose une théorie de moyenne portée plus limitée dans un contexte particulier celui des TPO. Il me sera possible de la tester empiriquement par le biais d'hypothèses que je présente à la section 1.2.

BIBLIOGRAPHIE

Adams, Peter, 1993. The title of the work. In : *The name of the journal*. juillet 1993. Vol. 4, n° 2, p. 201-213.

Babington, Peter, 1993. *The title of the work*. 3e éd. The address : The name of the publisher. 10 ser. ISBN 3257227892.

Caxton, Peter, 1993. *The title of the work*. juillet 1993. The address of the publisher : How it was published.

Draper, Peter, 1993. *The title of the work*. juillet 1993. The address of the publisher : The organization ; The publisher. 5 ser.

Eston, Peter, 1993. The title of the work. In :. 3e éd. The address of the publisher : The name of the publisher. 5 ser. p. 201-213.

Farindon, Peter, 1993. The title of the work. In : EDITOR, The (éd.), *The title of the book*. 3e éd. The address of the publisher : The name of the publisher. 5 ser. p. 201-213.

Gainsford, Peter, 1993. *The title of the work*. 3e éd. The address of the publisher : The organization.

Harwood, Peter, 1993. *The title of the work*. Mémoire de master. The address of the publisher : The school of the thesis.

Isley, Peter, 1993. *The title of the work*. juillet 1993. How it was published.

Joslin, Peter, 1993. *The title of the work*. Thèse de doctorat. The address of the publisher : The school of the thesis.

KIDWELLY, Peter (éd.), 1993. *The title of the work*. The address of the publisher : The organization ; The name of the publisher. 5 ser.

Lambert, Peter, 1993. 2 : *The title of the work*. The address of the publisher. The institution that published.

Marcheford, Peter, 1993. *The title of the work*.