UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ANALYSE COMPARATIVE DES MODÈLES DE MAINTENANCE DU LOGICIEL ENTRE ISO/IEC 14764, SWEBOK ET LES TRAVAUX DE PIGOSKI

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLLE
DE LA MAÎTRISE EN INFORMATIQUE DE GESTION

PAR
ASMA SELLAMI

AVRIL 2001

Mémoire approuvé par :

Alain Abran,
Professeur au Département d'informatique
Directeur au Laboratoire de recherche en gestion des logiciels

Robert Dupuis,
Professeur au Département d'informatique
Directeur au programme de l'École de Technologie de l'Information

REMERCIEMENTS

Je tiens à remercier mes directeurs de recherche, Messieurs Alain Abran et Robert Dupuis, pour leurs supports enthousiastes et pour leurs conseils judicieux qu'ils m'ont donnés durant mes travaux

Je tiens à remercier également tous les membres du Laboratoire de recherche en gestion des logiciels, notamment à Michèle Hébert pour les bons conseils concernant la présentation graphique et la qualité linguistique du texte.

Mes reconnaissances à toutes les personnes qui m'ont aidé à la réalisation de mon mémoire dans le cadre de ma maîtrise en informatique de gestion.

À mes parents qui n'ont jamais manqué d'amour,
d'affection et d'encouragement.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Nous nous intéressons aux modèles de maintenance du logiciel dans un contexte de l'ingénierie. Ces modèles ne sont pas conçus de la même façon entre les académiciens et les gens de l'industrie.

Notre travail consiste à réaliser une analyse comparative des divers modèles de maintenance entre ISO/IEC 14764, SWEBOK (version 0.7) et les travaux de Pigoski selon les critères de Vincenti.

Notre objectif principal est d'apporter des suggestions d'amélioration aux modèles de maintenance de ISO/IEC 14764 et de SWEBOK. Afin d'atteindre cet objectif, nous avons utilisé les critères de Vincenti des connaissances de conception de l'ingénierie comme étant des critères de comparaison. En se basant sur ces critères, nous avons identifié les modèles de maintenance les plus près des pratiques matures de l'ingénierie. Nous avons identifié également les forces et les faiblesses de ces modèles.

Cette étude nous a permis de conclure que les modèles de maintenance ne sont pas encore matures. De plus, les terminologies ne sont pas tous uniformes dans la littérature et dans les normes, et il y a une divergence sur la façon de considérer la maintenance. Notre recherche nous a permis de fournir des recommandations d'amélioration aux modèles de ISO/IEC 14764 et de SWEBOK. Ce travail pourra servir de point de départ pour d'autres recherches sur le sujet afin de continuer à améliorer le domaine de maintenance du logiciel et la discipline du génie logiciel en général.

MOTS CLÉS : Modèles de maintenance, ISO/IEC 14764, SWEBOK, Critères de Vincenti, Pigoski.

ABSTRACT

Our center of interest is focused on software maintenance models in context of engineering. These models have been designed from various perspectives by academicians and practitioners.

Our purpose was to review and analyse some models and identify areas where they could be improved. In this work we used the Vincenti's criteria of engineering knowledge to design and conduct various comparisons maintenance models in the public scope, such as the ones in ISO/IEC 14764 and SWEBOK (version 0.7).

This analysis highlighted that these maintenance models were not mature yet, for example: until now, the maintenance terminology is not only uniform, but also there wasn't a convergence in the way of maintenance consideration. Thus, our research allowed us to identify some recommendations in order to improving these ISO/IEC 14764 and SWEBOK models. This study can be used as a basis for further researches to continue and improve the maintenance knowledge area in particular, and software engineering in general.

KEY WORDS : Maintenance Models, ISO/IEC 14764, SWEBOK, Vincenti's criteria, Pigoski.

INTRODUCTION

La maintenance du logiciel est une fonction importante dans le cycle de vie du logiciel. Elle consomme entre 65 et 75% de l'effort total d'un logiciel selon différentes sources [Arthur, 1988; Swanson et Beath, 1989; Sharpe et al., 1991; Somerville, 2001]. Les coûts alloués à la maintenance augmentent de plus en plus malgré les efforts réalisés dans le domaine du génie logiciel couvrant les différentes activités du cycle de vie du logiciel [Pigoski, 1997]. Parmi les raisons que l'on peut donner à ces coûts élevés est le manque de concordance entre ce qui est présenté dans la littérature et ce qui est appliqué dans la pratique.

Plusieurs modèles ou documents de synthèse sur la maintenance ont été proposés dans la littérature pour aider à mieux gérer la maintenance, et donc indirectement pour en diminuer les coûts. Selon la norme ISO/IEC 14764, ces modèles sont conçus de point de vue de gestion. D'autre part, dans le projet SWEBOK (le guide de corpus de connaissance en génie logiciel), c'est l'aspect connaissance qui est mis en relief [SWEBOK, 2000].

L'objectif principal de notre analyse consiste à apporter des suggestions d'amélioration des modèles de maintenance et le génie en général, particulièrement, ceux de la norme ISO/IEC 14764 et du projet SWEBOK (Version 0.7). Pour ce faire, nous cherchons ainsi à présenter, dans un contexte de l'ingénierie, un nouveau rapport comportant aussi bien le modèle de maintenance de ISO/IEC 14764 amélioré que celui de SWEBOK amélioré.

Pour rendre possible la discussion cohérente, et aussi le consensus quant à la description de ce qui est perçu et constaté entre les académiciens et les gens de l'industrie, nous avons orienté notre recherche à la réalisation d'une analyse comparative des divers modèles de connaissance de maintenance. En raison de l'éclatement contemporain du savoir en une multitude de disciplines variées, et aussi de la complémentarité de ces disciplines, il serait utile de profiter des informations qui nous proviennent de d'autres disciplines telles que dans notre cas le génie aéronautique. De plus, notre analyse doit se baser sur des critères de comparaison afin de comparer et de juger la maturité des modèles de maintenance.

M. Walter G. Vincenti a défini, dans le domaine du génie aéronautique, des critères sur la classification des connaissances de conception de l'ingénierie. Notre travail consiste à appliquer ces

critères dans le domaine de maintenance du logiciel pour mieux clarifier et identifier les modèles de maintenance les plus près des pratiques matures de l'ingénierie, identifier les forces et les faiblesses de ces modèles, et en proposer des suggestions d'amélioration.

Dans le cadre de cette recherche, un procédé exploratoire fournit une base pour l'avancement utile en connaissance et en compréhension. Puisque le domaine de maintenance du logiciel n'est pas encore mature, il est certainement candidat pour les méthodes d'analyse exploratoire. Ces méthodes seront utilisées pour aider à mieux comprendre, contrôler et améliorer les modèles de connaissance de la maintenance du logiciel.

Dans le premier chapitre, nous présenterons la problématique, la motivation, la portée et l'objectif principal ainsi que la méthode de Basili[1] que nous avons adoptée pour la mise en œuvre de notre projet. Nous présenterons également les critères de Vincenti et leurs utilités dans le domaine de la maintenance des logiciels. Dans le deuxième chapitre, nous présenterons le projet SWEBOK afin d'identifier les forces et les faiblesses ainsi que la maturité de son modèle de maintenance Version (0.7). Cette identification est basée sur les critères de Vincenti. De même, l'analyse des modèles de Pigoski et ISO/IEC 14764 sera présentée respectivement dans les chapitres 3 et 4. Le dernier chapitre présentera une étude comparative de ces modèles. Cette étude d'évaluation de maturité est basée, bien entendu, sur les critères de Vincenti. De plus, nous présenterons des suggestions d'amélioration de ces modèles et en particulier ISO/IEC 14764 et SWEBOK (version 0.7).

Nous terminerons ce travail par une conclusion qui porte essentiellement sur la maturité des modèles de maintenance, l'interprétation des résultats d'analyse obtenus, et les perspectives futures de la maintenance en tant que discipline de l'ingénierie.

---

[1] Cadre de Basili modifié par Abran et al. (1999).

CHAPITRE I

PRÉSENTATION DU TRAVAIL

1.1 PROBLÉMATIQUE DE RECHERCHE

La maintenance du logiciel est définie dans la norme IEEE [IEEE 729, 1983] comme l'ensemble des travaux réalisés sur un logiciel après sa mise en opération. Historiquement, elle se réalise lors de l'apparition d'un problème ou suite à une demande de modification. Cependant, des travaux récents ont suggéré que les activités de maintenance doivent être considérées et prises en compte aussi durant le cycle de développement d'un logiciel [Somerville, 2001 ; ISO/IEC 14764].

Selon une étude réalisée auprès des entreprises américaines, 30 milliards US $ ont été dépensés annuellement pour la maintenance du logiciel dans les années 1990, et 90% des coûts du cycle de vie d'un logiciel est affectée à la maintenance [Pigoski, 1997].

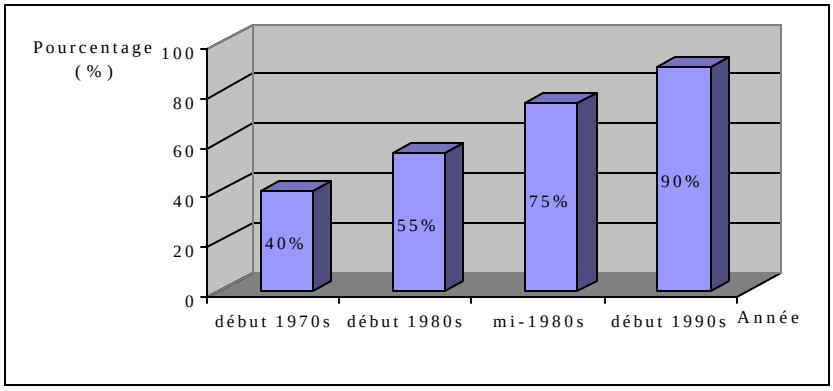

Figure 1.1 : Coûts de la maintenance du logiciel [Pigoski, 1997]

À partir de cette figure 1.1, on peut constater que les coûts consacrés à l'activité de maintenance du logiciel augmentent malgré les efforts réalisés dans le domaine du génie logiciel couvrant les

différentes activités du cycle de vie du logiciel. Parmi toutes les explications que l'on peut donner à ces coûts élevés, on peut citer par exemple l'inexistence ou la mauvaise qualité de la documentation, ce qui fait que les mainteneurs consomment beaucoup de temps dans la compréhension du code.

Dans la littérature, plusieurs auteurs ont proposé divers modèles de maintenance du logiciel, incluant certains modèles globaux. Néanmoins, le problème réside dans le fait qu'il n'y a pas une pleine concordance entre ce qui est présenté dans la littérature et ce qui est appliqué dans la pratique. En particulier, les modèles ou les documents de synthèse sur la maintenance de la norme ISO/IEC 14764 ont été développés par discussion d'experts, mais n'ont pas été vérifiés par des observations systématiques dans un large éventail de contextes industriels ni par d'autres que les participants à la création de ces modèles. Ces documents prescrivent ce qu'un mainteneur doit faire plutôt que la diversité des choses que ce mainteneur peut ou pourrait faire.

La maintenance du logiciel est un domaine mal compris du génie logiciel et relativement peu d'études portent sur la maintenance, que ce soit du point de vue pratique ou académique. Ce manque d'études contribue à une faible compréhension des gens sur la maintenance [Pigoski, 1997].

Il existe plusieurs définitions de la maintenance du logiciel :

- La maintenance du logiciel est la modification d'un produit logiciel après sa livraison pour en corriger les erreurs, améliorer la performance ou d'autres attributs ou adapter le produit à un environnement modifié [IEEE 1219],
- La maintenance signifie l'évolution des logiciels [Sommerville, 1996],
- La maintenance est décrite comme étant le processus d'un produit logiciel qui subit « la modification du code et les documents associés suite à l'apparition d'un problème ou un besoin d'amélioration. L'objectif consiste à modifier le produit logiciel existant tout en gardant son intégrité » [ISO/IEC 12207],
- La maintenance du logiciel est définie par la norme ISO/IEC 14764 de la même façon que la norme ISO/IEC 12207 tout en mettant l'accent sur les aspects pré-livraison de maintenance tels que par exemple : le planning [ISO/IEC 14764],
- La maintenance du logiciel est décrite par les chercheurs et les praticiens comme étant la totalité des activités requises pour fournir un support efficace à un système logiciel. Ces activités sont exécutées aussi bien durant les étapes pré-livraison qu'après la livraison. Les activités pré-livraison comportent le planning pour les opérations post-livraison, le support et la détermination logistique. Par contre les activités post-livraison comportent la modification du logiciel, la formation et les opérations de support [SWEBOK, 2000].

D'autre part, selon la norme ISO/IEC 14764, la qualité du logiciel est une considération importante dans la maintenance du produit logiciel, mais la qualité en soi peut avoir plusieurs significations.

Malgré des terminologies différentes, les mainteneurs devraient avoir un programme de qualité de logiciel répondant aux caractéristiques de qualité de la norme ISO [ISO/IEC 9126]. Un processus doit être implanté afin d'identifier, définir, sélectionner, appliquer, valider et améliorer la mesure du logiciel pour la maintenance. Entre autres activités, le gestionnaire de la maintenance doit déterminer l'effort (en terme de ressources dépensées) pour les différents types de maintenance. Les données doivent être collectées, analysées et interprétées afin de faciliter l'amélioration du processus de maintenance du logiciel et afin d'obtenir une meilleure compréhension de la répartition et l'estimation des coûts de maintenance.

De plus, on retrouve plusieurs problèmes qui sont liés à la maintenance. Selon Pfleeger, ces problèmes peuvent être d'ordre technique et/ou de gestion [Pfleeger, 1998].

1.1.1 Problèmes techniques

Compréhension limitée : des recherches dans la pratique révèlent que 40 à 60% de l'effort de maintenance est dévoué à la compréhension du code à modifier [Pfleeger, 1998; Takang et Grubb, 1997; Dorfman et Thayer, 1997]. La compréhension du code logiciel représente un défi pour les mainteneurs. En effet, il est souvent difficile d'expliquer la progression d'un code à travers ses versions car les modifications apportées ne sont pas documentées.

Test : la réalisation des tests peut être considérée comme étant un problème lorsqu'il n'y a pas assez de temps pour tester. Par exemple, le système de réservations aériennes devrait toujours être disponible. Cependant, il est difficile de convaincre les utilisateurs à céder leurs systèmes pour des raisons de tests. Quand un système exécute une fonction critique telle que le contrôle de trafic aérien ou une intervention médicale d'un patient, il est presque impossible d'arrêter le système afin de le tester. Dans ces cas de figure, les tests seront exécutés sur des systèmes dupliqués; ensuite les changements seront transférés aux systèmes de production. Mais, les coûts d'exécution des tests peuvent être considérables en terme de temps et d'argent.

Analyse de l'impact : un produit logiciel doit subir une analyse de l'impact afin de minimiser le risque et d'évaluer l'impact de changement à un nouvel environnement. Ainsi, les habiletés du mainteneur et la complétude des documents sont utiles dans ce domaine.

Maintenabilité : la norme IEEE [IEEE 610.12] définit la maintenabilité comme étant la facilité selon laquelle un produit logiciel peut être maintenu pour satisfaire aux exigences spécifiés. Il est important d'introduire les caractéristiques de maintenabilité dans l'effort de développement du logiciel afin de réduire les coûts du cycle de vie du logiciel et améliorer la qualité de la maintenance. Mais, dans la pratique les caractéristiques de maintenabilité ne figurent pas dans le processus de développement du logiciel. Ce qui constitue un problème pour les mainteneurs. Si les normes sont respectées dans la pratique, les problèmes de maintenance peuvent être évités [Dorfman et Thayer, 1997].

1.1.2 Problèmes de gestion

Priorités de gestion : L'équipe de maintenance doit tenir compte des exigences des clients du point de vue gestion [Pfleeger, 1998]. En effet, dans la pratique les priorités de gestion sont considérées plus importantes que les priorités techniques et la plupart des compagnies se concentrent sur les systèmes existants plutôt que d'investir dans de nouvelles alternatives. Cependant, comme les gestionnaires encouragent les mainteneurs à améliorer les vieux systèmes, il est nécessaire d'adapter les utilisateurs à des nouvelles fonctionnalités dans le système. D'autre part, la précipitation de lancer rapidement un produit ou une nouvelle fonction sur le marché, peut conduire les développeurs ou les mainteneurs à réaliser un produit inélégant, pauvrement testé plutôt que de prendre suffisamment le temps pour suivre les bonnes pratiques du génie logiciel. D'où les problèmes de compréhension limitée par les mainteneurs.

Moral : Des études réalisées par Lietz [Lietz et Swanson, 1981] ont montré que 11,9 % des problèmes durant la maintenance résultent d'une productivité et d'un moral faible. La raison majeure pour le faible moral est le statut de classe secondaire accordée souvent à l'équipe de maintenance. Les mainteneurs s'occupent des problèmes que les développeurs n'ont jamais pris en considération. Les mainteneurs devraient être qualifiés non seulement pour écrire le code mais aussi pour travailler avec les utilisateurs dans l'anticipation des changements et dans les limites. Ainsi, les mainteneurs nécessitent d'avoir des capacités lucides et de persévérances pour comprendre les opérations internes d'un système pour améliorer le code et les documents associés.

Dans certaines organisations, certains membres de l'équipe de maintenance accomplissent leur fonction à tour de rôle dans plusieurs projets pour faire le maximum dans un intervalle de temps limité. Cette rotation peut engendrer des problèmes de conflits de priorités. Lors de la réalisation de la

maintenance d'un système, 8% des problèmes résultent de l'appartenance des mainteneurs dans plusieurs directions [Pfleeger, 1998].

Tous ces problèmes sont dus au fait qu'il n'y a pas assez de concordance entre la théorie et la pratique. Les modèles de maintenance sont développés par discussions entre des experts, mais ne sont pas vérifiés par d'autres que les participants ou par de l'expérimentation en contextes industriels.

1.2 MÉTHODE DE LA RECHERCHE

La méthode à suivre dans cette étude est basée sur le cadre d'expérimentation proposé par Basili [Basili et al., 1986]. Ce cadre conceptuel a été adapté pour son application dans les études de type exploratoire par le Laboratoire de recherche en gestion des Logiciels de l'Université du Québec à Montréal. Ce laboratoire se spécialise principalement en études dans le domaine du génie logiciel, comme par exemple les recherches sur les principes fondamentaux du génie logiciel et sur la mesure de la taille fonctionnelle des logiciels.

Ce cadre a été élaboré initialement aux cas des études de type expérimental pour fournir un protocole de recherche simple, bien structuré et facile à utiliser. Il comporte essentiellement quatre phases de déroulement d'une recherche : la définition, la planification, l'opération et l'interprétation.

À l'origine, ce cadre de recherche de Basili [Basili et al., 1986] a été utilisé dans les domaines de recherche matures où le corpus de connaissance est déjà structuré. Cependant, dans les domaines en émergence où le corpus de connaissance n'est pas encore structuré, il est nécessaire d'apporter une certaine adaptation à ce cadre. Cette adaptation a été réalisée principalement pour documenter et réaliser des projets de recherche exploratoire [Abran et al., 1999].

Le cadre adapté pour le cas des recherches exploratoires est composé de quatre phases principales : la définition, la planification, l'opération, et l'interprétation des résultats. Chacune de ces phases est composée de plusieurs étapes. Chaque étape comporte plusieurs activités nécessaires au déroulement de notre projet. Un modèle de ce cadre est présenté dans la figure 1.2. Un résumé du cadre dûment rempli est aussi présenté dans l'appendice A.

Tableau 1.1 : Cadre de Basili adapté à la recherche exploratoire par [Abran et al.]

1. Définition			
Motivation	Portée	Objectif	Utilisateurs de recherche
2. Planification			
Étapes du projet	Intrants du projet		Livrables du projet
3. Opération			
Analyse des documents	Feedback des praticiens		Modèle proposé
4. Interprétation			
Contexte d'interprétation	Extrapolation des résultats		Travaux futurs

1.3 DÉFINITION DU PROJET

Dans cette section, nous présentons le contenu de la première phase de Basili modifié permettant de définir les objectifs et les limites de cette recherche.

La motivation principale de ce travail de recherche consiste à aider à améliorer les modèles de maintenance du logiciel du point de vue de l'ingénierie.

La portée de l'étude est les modèles ISO/IEC 14764 et SWEBOK portant sur la maintenance du logiciel.

L'objectif principal de notre recherche consiste à proposer des suggestions d'amélioration aux modèles de maintenance du logiciel de la norme ISO/IEC 14764 et de SWEBOK (version 0.7). Pour cela, nous avons à effectuer une analyse comparative des domaines de connaissance de ces modèles en s'aidant à partir des critères de Vincenti et des travaux de Pigoski.

Cette comparaison offrira la possibilité d'identifier les désaccords entre SWEBOK et la norme ISO/IEC 14764 sur la maintenance pour les améliorer.

La question de recherche que nous posons est donc la suivante :

- Existe-t-il des divergences entre SWEBOK, ISO/IEC 14764 et les travaux de Pigoski sur la façon de considérer la maintenance des logiciels en se basant sur les critères de Vincenti?

Cette question peut être subdivisée en deux sous-questions :

1. Est-ce que les connaissances généralement reconnues par SWEBOK sont compatibles avec ce qui se trouve dans les normes?
2. Si des faiblesses apparaissent dans les connaissances sur la maintenance telles qu'évaluées à l'aide des catégories de Vincenti, trouve-t-on dans les travaux de Pigoski de quoi combler ce vide?

Utilisateurs des résultats de la recherche : Notre projet de recherche a d'intérêt pour de nombreuses personnes dont les activités sont liées d'une façon ou d'une autre à la maintenance du logiciel :

- Les chercheurs du projet SWEBOK qui pourront profiter d'une liste de principes importants dans le domaine de connaissance lié à la maintenance.
- Les développeurs de normes en maintenance qui pourront disposer des suggestions d'amélioration aux modèles de maintenance.
- Les spécialistes dans le domaine de maintenance en génie logiciel qui pourront bénéficier des éléments contenus dans cette étude et trouver également de nouvelles pistes de recherche ainsi que des perspectives d'avenir.

1.4 PLANIFICATION DU PROJET DE RECHERCHE

Afin de réaliser les livrables finaux de notre recherche, nous avons identifié les étapes suivantes :

1. Nous procédons à une étude des critères de Vincenti pour les utiliser comme des critères de comparaison. Ces critères sont au nombre de six :
    1. Les concepts fondamentaux de conception,
    2. Les critères et les spécifications,
    3. Les outils théoriques,
    4. Les données quantitatives,
    5. Les considérations pratiques,
    6. Les instruments de conception.

2. Nous analysons les divers modèles de maintenance selon les critères de Vincenti identifiés pour dégager leurs forces et leurs faiblesses.

Selon Fenton [Fenton et Pfleeger, 1997], un modèle est une abstraction de la réalité nous permettant de dévoiler les détails et d'étudier une entité ou un concept selon une perspective particulière. Comme exemple, les modèles de coût qui nous permettent d'examiner seulement les aspects d'un projet contribuant au coût final. Les modèles peuvent être sous différentes formes tels que par exemple, les équations, les transformations «*mappings*» ou les diagrammes. Ceci nous montre comment les composants sont liés les uns aux autres et ainsi nous pouvons examiner et comprendre leurs relations et fournir des jugements.

Dans cette étude nous nous intéressons aux trois modèles de maintenance :

- SWEBOK : swebok knowledge area description for software maintenance (Version 0.7),
- Travaux de Pigoski : *Practical Software Maintenance : Best Practices for Managing Your Software Investment* de Pigoski (1997),
- Norme : ISO/IEC 14764.

3. Nous analysons chacun de ces modèles en suivant les étapes suivantes :

- Classification du modèle de maintenance spécifié selon les critères de Vincenti déjà identifiés,
- Vérification si le modèle spécifié répond à tous les critères identifiés,
- Identification des forces et des faiblesses qui peuvent exister dans ce modèle.

4. Étude comparative des modèles de maintenance de SWEBOK, de la norme ISO/IEC 14764 et de les travaux de Pigoski.

- Identification des désaccords qui peuvent exister entre ces différents modèles.

5. Dans la dernière étape de la recherche, nous procédons à une préparation des recommandations des modèles de maintenance de SWEBOK et ISO/IEC 14764.

Concernant l'étape 3 « Opération » du cadre de Basili, elle sera présentée dans les chapitres 3, 4 et 5. L'étape 4 « Interprétation » est abordée dans le chapitre 5 et la conclusion.

1.5 LIMITE DE LA RECHERCHE

La recherche n'étudiera que les modèles de maintenance du logiciel du point de vue de l'ingénierie, particulièrement de la norme ISO/IEC 14764 et du projet SWEBOK pour aboutir à des recommandations d'amélioration de ces modèles.

D'autre part, afin de combler le vide qui pourra exister dans les connaissances sur la maintenance telles qu'évaluées à l'aide des critères de Vincenti, nous prévoyons l'utilisation des livres indiqués ci-dessous :

- Designing Maintainable Software [Smith, Dennis D., 1999],
- Software Engineering : Theory and Practice [Pleeger, Shari L., 1998],
- Software Maintenance [Takang, Armstrong A. et Grubb Penny A., 1997],
- Maintaining Information Systems in Organisations [Swanson, E. Burton et Beath, C. Mathis, 1989].
- Software Engineering : A Practitioner's Approach [Pressmann, Roger S. McGraw Hill, 1987].

Il est clair que cette recherche ne comporte pas la gestion de la maintenance d'une entreprise ou d'un logiciel spécifique tant au niveau de la conception qu'à celui de l'implantation. Il s'agit d'effectuer une analyse comparative, selon les critères de Vincenti, du contenu d'un domaine de connaissance en ingénierie, particulièrement la maintenance du logiciel du point de vue de l'ingénierie et non pas du point de vue strictement de codification.

1.6 CRITÈRES DE VINCENTI

1.6.1 Utilité des critères de Vincenti

Notre motivation de considérer les critères de Vincenti [Vincenti, 1990] consiste à mieux comprendre, dans un contexte de l'ingénierie, les sous-domaines de connaissance liés à la maintenance du logiciel et à mieux identifier les modèles de maintenance les plus près des pratiques matures d'ingénierie. En effet, pour un domaine d'ingénierie mature, tous les éléments identifiés selon les critères de Vincenti seront présents. À l'opposé, dans un domaine de connaissances en émergence, seulement quelques éléments seraient présents (partiellement ou complètement). De plus, l'utilisation de ces critères nous a permis de proposer des suggestions d'amélioration des modèles de maintenance du logiciel.

Dans son livre « *What Engineers Know and How They Know It – Analytical Studies from Aeronautical History* », M. Vincenti examine l'évolution des connaissances dans le domaine du génie aéronautique. En premier lieu, il nous a présenté un certain nombre d'études de cas historiques traitant plusieurs problèmes liés à la conception des avions dans la période des années 1908 et 1953. Ces problèmes concernent en particulier la conception des ailes d'un avion, le développement de la théorie d'analyse contrôle-volume, etc.

En second lieu, nous avons remarqué qu'un chapitre principal intitulé « *The Anatomy of Engineering Design Kowledge* » montre ce que Vincenti propose comme structure des connaissances du génie en général. Dans ce chapitre, on retrouve une tentative de catégorisation des critères de chaque connaissance basée essentiellement sur des études de cas et de l'expérience propre de l'auteur sur le génie. Bien que Vincenti admet qu'une telle généralisation présente un risque puisqu'elle provient du domaine du génie aéronautique, il croit qu'elle peut être universellement applicable (Réf. Vincenti, p. 200). Les six catégories ou critères des connaissances du génie qu'il propose sont les suivantes :

1. Les concepts fondamentaux de la conception : Ces concepts consistent à des principes opérationnels associés à des configurations normales des systèmes à concevoir. Les principes opérationnels et les configurations normales sont deux aspects qui définissent la technologie normale et l'approche de conception, contrairement à la conception radicale, où des nouveaux concepts et des techniques sont nécessaires. En ce sens, Vincenti énonce que ces concepts peuvent exister seulement d'une manière implicite dans l'esprit du concepteur. Ils sont acquis avant d'identifier les spécifications d'un projet. Ils sont appris par les ingénieurs lors de leurs formations académiques avant d'entrer formellement dans l'expérience de l'ingénierie.

Le principe opérationnel et la configuration normale fournissent un cadre dans lequel une conception normale prend place. La traduction de ces concepts dans une conception plus concrète nécessite la connaissance des catégories qui suivent.

2. Les critères et les spécifications : Ce type de connaissance permet à l'ingénieur ou au concepteur du système d'identifier les besoins spécifiques en terme matériel et de traduire les buts qualitatifs d'un système (le général) en des buts quantitatifs cachés dans des termes techniques concrets (le spécifique). Pour accomplir leurs tâches, les concepteurs devront avoir des connaissances sur les critères techniques appropriés d'un système et de son utilisation ainsi que les limites de ces critères techniques. Ainsi, sans ces spécifications techniques, il est presque difficile pour un concepteur, de fournir les

détails et les dimensions nécessaires aux développeurs. À cet effet, Vincenti considère que la connaissance fondamentale consiste à sélectionner un ensemble approprié de critères. Mais, les critères de conception varient largement selon la perception des concepteurs. Dans certains cas, ces critères sont simples et évidents comme par exemple la rapidité, l'altitude, etc. Dans d'autres cas, ces critères ne sont pas immédiatement clairs, ils doivent être divisés consciemment et délibérément sur certaines périodes.

En plus, la détermination des critères de base est dépendante des « outils théoriques » et des « données quantitatives ».

3. Les outils théoriques : Ces outils sont issus des modèles mathématiques et des théories ainsi que des concepts intellectuels. Les modèles mathématiques et les théories sont nécessaires pour aboutir à une analyse quantitative. Par contre, les concepts intellectuels sont des concepts qui traduisent la façon de raisonner et qui permettent d'aboutir à la conceptualisation qualitative (« des concepts pour penser à la conception »). De tels concepts et outils incluent la connaissance scientifique générale appelée par Vincenti « des théories phénoménologiques », des techniques *ad hoc*, approximation, ou des théories utiles seulement pour faire du calcul en génie.

Ces outils théoriques qui sont nécessaires à l'analyse conceptuelle comportent :

- Les outils mathématiques, pas de contenu physique,
- Les outils mathématiques, basés sur les théories scientifiques,
- Les outils mathématiques, basés sur les théories de l'ingénierie,
- Les outils mathématiques, basés sur des hypothèses,
- Les concepts intellectuels qui fournissent un langage de réflexion. Ceux-ci sont employés non seulement dans l'analyse quantitative mais également dans la conception qualitative.

4. Les données quantitatives : Données obtenues empiriquement selon les propriétés physiques d'un système, représentés typiquement dans des tableaux ou des graphes. Le rôle de telles données, qui peuvent être descriptives (comment les choses sont?) aussi bien que prescriptives (comment les choses devraient être?, par exemple, facteur de sûreté), consiste à déterminer les détails du système à concevoir.

5. Les considérations pratiques : Les considérations pratiques sont des connaissances basées sur les expériences de l'ingénieur ou sur des conventions. Généralement, ces connaissances ne sont pas formellement codifiées. Elles sont souvent représentées par des règles empiriques «rules of thumb ».

Selon Vincenti, quand ce type de connaissance devient formellement enregistré et bien codifié, il sera inclus facilement dans une autre catégorie ou un critère de connaissances.

6. Les instruments de conception : Ces instruments représentent les différentes approches et démarches de résolution d'un problème comme les procédures (exemple, décomposition hiérarchique), les chemins de pensées (incluant la pensée visuelle), les capacités de jugement et la connaissance du « Comment » accomplir les tâches.

Il est à noter que l'identification de chaque thème de connaissances en se basant sur les critères de Vincenti n'est pas une tâche facile. En effet, comme Vincenti le déclare, ces critères progressent dans leurs évolutions et ne sont pas mutuellement exclusives. En ce sens, les frontières entre ces catégories sont relativement confuses. Cependant, Tremblay, auteur de la version *Jump-Start* (SWEBOK) et spécialiste dans le domaine des connaissances de construction et de conception des logiciels, a présenté une classification dans les thèmes des domaines de connaissances de conception en utilisant les critères de Vincenti. De plus, Tremblay énonce que l'application de ces critères dans le domaine du génie logiciel est différente de celle du domaine du génie informatique où on utilise des concepts concrets.

Une classification concernant le domaine des connaissances de maintenance des logiciels basée sur les critères de Vincenti a été produite par T. Pigoski [SWEBOK, 2000] dans le cadre du projet SWEBOK (version 0.7). Cette classification fait l'objet du chapitre suivant.

CHAPITRE II

ANALYSE DU MODÈLE DE SWEBOK SELON LES CRITÈRES DE VINCENTI

Dans ce chapitre, nous allons décrire les fondements du projet SWEBOK que nous avons retenus. Par la suite, nous effectuerons une analyse du domaine des connaissances de maintenance qui se trouvent dans ce projet en se basant sur la description de Vincenti, des critères des connaissances de conception de l'ingénierie tout en présentant une synthèse critique sur chaque sous-domaine de maintenance.

2.1 LES FONDEMENTS DU PROJET SWEBOK

Le projet SWEBOK, guide au corpus des connaissances en génie logiciel, est commandité par l'IEEE Computer Society et un certain nombre de corporations et d'organisations publiques. Il est géré par une équipe de l'UQAM. Les documents de ce projet sont disponibles sur Internet (www.swebok.org).

2.1.1 Objectifs du projet SWEBOK

Les principaux objectifs du projet SWEBOK sont les suivants :

1. Décrire le contenu de la discipline du génie logiciel;
2. Donner un accès selon les sujets, au corpus des connaissances du génie logiciel;
3. Promouvoir à l'échelle mondiale une vue cohérente sur le génie logiciel;
4. Définir la place et tracer les frontières du génie logiciel relativement à d'autres disciplines comme l'informatique, la gestion de projets, le génie informatique et les mathématiques ;
5. Poser les fondations du cursus et de l'octroi de certificats de compétence professionnelle.

2.1.2 Organisation hiérarchique

Essentiellement, le projet SWEBOK vise à développer un consensus sur le noyau des connaissances caractérisant la discipline du génie logiciel. Afin de réaliser ce but, les dix principaux domaines de connaissance ont été identifiés. Les descriptions de ces domaines ont été développées et révisées par un nombre considérable d'experts internationaux (par exemple, dans le domaine de la maintenance, plus que 40 personnes ont révisé la version 0.5 et dans tous les domaines de connaissance, il y a respectivement 378 et 394 révisions sur les versions Stoneman 0.5 et 0.7 du projet). Chaque

description d'un domaine de connaissance comporte, entre autres, une définition du domaine de connaissance, l'analyse et la description des sujets « généralement reconnues » du domaine de connaissance, une liste du matériel de référence recommandée et une classification de certains sujets des domaines de connaissance basées principalement sur les critères de Vincenti de la classification des connaissances de conception de l'ingénierie.

Ainsi, ce projet utilise une organisation hiérarchique pour décomposer chaque domaine de connaissances. Cette organisation permet aux lecteurs de trouver aisément les sujets qui les intéressent. Le tableau de la page suivante, inspiré du guide SWEBOK (version 0.7), énumère les divers domaines de connaissance et indique les sous-domaines de connaissance correspondant à chacun.

Le traitement des sujets du guide se veut compatible avec celui des principales écoles de pensée et avec celui en usage dans l'industrie, dans la littérature technique et les normes relatives au génie logiciel. Ce traitement ne présuppose ni domaines d'application, ni utilisations commerciales, ni philosophies de management, ni méthodes de développement en particulier. La description de chacun des sujets se limite à celle qui est nécessaire au lecteur pour trouver les documents de référence.

2.1.3 Principe des connaissances généralement reconnues

Dans la version 0.5 du guide [Bourque, 1999], il a été adopté le concept des connaissances généralement reconnues. Ces connaissances sont différentes des connaissances avancées et de celles propres à la recherche (en fonction de leur maturité) ainsi que de celles considérées comme spécialisées (à partir de la généralité de leur application). Les connaissances généralement reconnues s'appliquent la plupart du temps à la plupart des projets et un large consensus permet de valider leur intérêt et leur efficacité. Cependant, cela ne signifie pas que les connaissances généralement admises sont applicables de façon uniforme à tous les travaux relevant du génie logiciel mais, plutôt, les ingénieurs en logiciels compétents devraient maîtriser ces connaissances et être en mesure de les appliquer selon les besoins spécifiques à chaque projet.

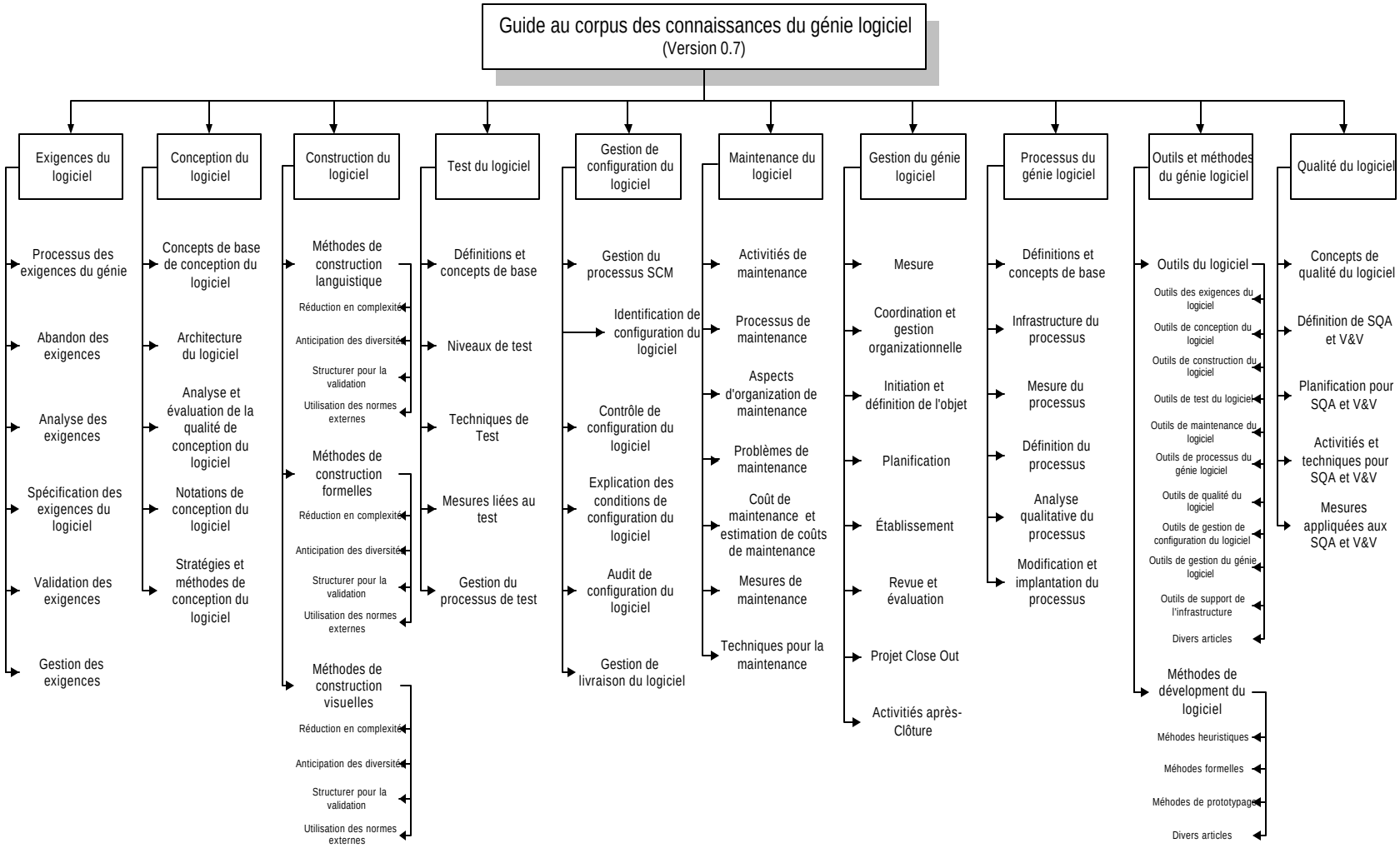

Figure 2.2 : Guide au corpus des connaissance du génie logiciel (Version 0.7)

En d'autres termes, ces connaissances sont des éléments qui devraient être étudiés en vue d'un examen conduisant à l'obtention d'un brevet en génie logiciel. Cependant, les connaissances acquises par les ingénieurs diffèrent selon les régions. En effet, le critère d'enseignement aux États-Unis ne s'applique pas nécessairement tels quels dans d'autres pays. Ainsi, les deux définitions des connaissances généralement admises sont considérées comme complémentaires.

2.2 ANALYSE DU MODÈLE DE SWEBOK SELON LES CRITÈRES DE VINCENTI

La classification d'un domaine de connaissances selon les critères ou les catégories de Vincenti n'est pas une tâche facile. En effet, plusieurs révisions de ces critères ont été réalisées, par Dr. Tremblay de l'Université du Québec à Montréal, pour classifier les connaissances de conception du projet SWEBOK (*Stone Man* Version 0.9). Dans le même contexte, Tomayko qui est expert de l'histoire en informatique et membre senior des personnels techniques au SEI (*Software Engineering Institute*), considère qu'en dehors du génie aéronautique, les critères de Vincenti deviennent difficiles à comprendre et peuvent être inapplicables (Communication privée). Par contre, selon Vincenti, ses critères sont généraux et peuvent être appliqués de façon universelle. Cependant, ces critères ne sont pas mutuellement exclusifs et leurs frontières, comme indiqué par Vincenti, sont relativement ambiguës ; certains éléments de connaissance d'un critère peuvent incorporer les caractéristiques de d'autres critères.

Nous cherchons à vérifier si les critères de Vincenti s'appliquent dans le domaine de maintenance du logiciel et d'une manière globale dans celui du génie logiciel.

Nous avons évoqué dans la section précédente que le projet SWEBOK vise à recueillir un consensus des domaines de connaissance du génie logiciel et que ce projet ne couvre pas tous les types des domaines de connaissance mais plutôt une partie, celles qui sont généralement admises. Dans ce contexte, seulement deux réviseurs de la version 0.5 de SWEBOK sur le domaine de maintenance du logiciel ont été interrogé pour donner leur propre classification :

- Thomas M. Pigoski qui est ingénieur senior en génie logiciel à TECHSOFT, l'auteur principal de la norme internationale sur la maintenance du logiciel proposée par ISO/IEC 14764, l'auteur d'un livre intitulé « Practical Software Maintenance : *Best Practices for Managing Your Software*

Investment, 1997 » que nous allons traiter dans le chapitre suivant et le président général des conférences internationales sur la maintenance du logiciel de la norme IEEE.

- Vinh Tuong Ho qui est professeur à l'Institut francophone d'informatique Hanoï, Vietnam.

Afin de classer les connaissances de maintenance selon les critères de Vincenti, Pigoski et Ho ont présenté chacun une décomposition selon leur propre analyse comme suit :

Tableau 2.2 : Classement des connaissances de maintenance selon les critères de Vincenti par Pigoski et Ho (Version 0.5 de SWEBOK).

Thomas M. Pigoski	Vinh Tuong Ho
1- LES CONCEPTS FONDAMENTAUX DE CONCEPTION	
Maintenance du logiciel : • Besoin pour la maintenance • Catégories de maintenance	Concept de maintenance Évolution du système
Activités de maintenance : • Activités uniques • Activités de support : gestion de configuration, qualité • Activité de planification de maintenance	Types des activités de maintenance Processus de maintenance
Aspect d'organisation de maintenance : • Mainteneur • Impartition • Structure organisationnelle	Aspect d'organisation de maintenance Problème de maintenance de logiciel Coût de maintenance
2- LES CRITÈRES ET LES SPÉCIFICATIONS	
Mesures de maintenance du logiciel : • Établissement d'un programme de mesure • Mesures spécifiques	Mesures de maintenance Coût d'estimation de maintenance
3- LES OUTILS THÉORIQUES	
	Outils de maintenance automatisés Gestion de configuration Rétro-conception Réingénierie Redocumentation
4- LES DONNÉES QUANTITATIVES	
Coût de maintenance et estimation du coût de maintenance • Coût • Estimation du coût : modèles paramétriques, expérience	

Thomas M. Pigoski	Vinh Tuong Ho
5- LES CONSIDÉRATIONS PRATIQUES	

Problèmes de maintenance du logiciel :
• Problèmes techniques :
  • Compréhension limitée
  • Test
  • Analyse de l'impact
  • Maintenabilité
	Considérations pratiques

• Problèmes de gestion :
  • Alignement avec les problèmes organisationnels
  • Personnel
  • Problème de processus
Techniques pour la maintenance :
• Compréhension du programme
• Réingénierie
• Rétro-conception
• Analyse de l'impact
Ressources	

6- LES INSTRUMENTS DE CONCEPTION	
Processus de maintenance : • Modèles du processus de maintenance	Modèles du processus de maintenance

Il découle des deux tableaux précédents qu'il n'existe pas de couverture totale des connaissances de maintenance par les critères de Vincenti. Nous avons remarqué que selon Pigoski, le troisième critère des « outils théoriques » n'est pas déterminé. D'où l'apparition des faiblesses dans ce critère et que les connaissances de ce type de maintenance ne sont pas suffisantes et assez matures. Quant à Ho, il néglige le quatrième critère des « données quantitatives » retenu par Pigoski et il indique qu'il y a plutôt un manque d'éléments liés au critère des «données quantitatives » (Swebok Jump -Start Documents). Dans les deux cas, aucune explication sur la classification n'a été donnée par ces auteurs. Ces deux différentes classifications présentées ci-dessus par Pigoski et Ho ne sont pas complémentaires l'une de l'autre. Étant donné que Pigoski est l'auteur du chapitre sur la maintenance et qu'il est expert dans le domaine, nous adoptons sa classification.

Au niveau du projet SWEBOK, un résumé de l'analyse du domaine de maintenance selon Pigoski est présenté dans la figure 2.3 :

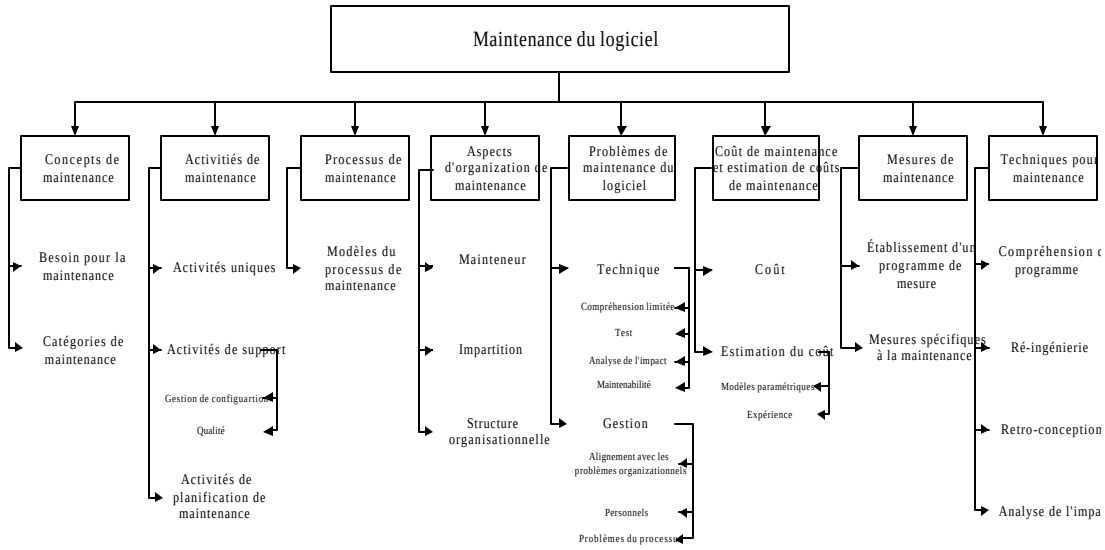

Figure 2.3 : Domaine de maintenance du logiciel (SWEBOK, Version 0.7)

Comme nous l'avons évoqué dans le chapitre précédent, notre démarche consiste à entreprendre une étude permettant d'analyser le domaine de connaissance de maintenance du point de vue de l'ingénierie pour dégager les forces et les faiblesses qui peuvent exister. Cette étude est basée sur la décomposition et l'analyse de Pigoski portant sur la maintenance. Dans les lignes qui suivent, nous décrivons en détail le contenu du chapitre de Pigoski selon les critères de Vincenti.

2.2.1 Les concepts fondamentaux de conception

Selon la classification présentée dans le tableau 2.2, Pigoski considère dans le critère « Concepts fondamentaux de conception » les éléments suivants :

1. Maintenance du logiciel : Pigoski spécifie le besoin de maintenir un système ainsi que les différents critères de maintenance du logiciel. Il montre l'utilité de la maintenance pour s'assurer que le système répond d'une façon continue aux exigences des utilisateurs. La maintenance devrait être accomplie afin de :

- corriger les défauts,
- corriger les flux conceptuels,
- avoir une interface avec d'autres systèmes (nouveaux ou modifiés),
- apporter des améliorations,

- apporter des modifications nécessaires à un système,
- apporter des modifications dans des fichiers ou des bases de données,
- améliorer la conception,
- convertir des programmes pour pouvoir bénéficier des avantages matériels et logiciels ainsi que les caractéristiques de télécommunication.

Plusieurs études ont été réalisées pour classifier les types de maintenance. Celle de Lehman [Lehman, 1986] considère la maintenance comme un développement continu sous la contrainte d'un système logiciel existant. Swanson fut le premier chercheur qui a examiné ce qui se passe réellement dans l'évolution et la maintenance du logiciel, en se basant sur des données empiriques puisées de la pratique industrielle [Swanson, 1989]. Il a regroupé la fonction de maintenance en trois catégories différentes selon les causes constatées : maintenance corrective, maintenance adaptative et maintenance perfective.

- **Maintenance corrective** : modification réactive à un produit logiciel réalisée après sa livraison pour en corriger les défauts.
- **Maintenance adaptative** : modification d'un produit logiciel réalisée après sa livraison pour garder le code utilisable suite à un changement ou à un environnement modifié.
- **Maintenance perfective** : modification d'un produit logiciel réalisée après sa livraison pour améliorer la performance ou la maintenabilité.

En plus de ces trois catégories de maintenance, les normes IEEE [IEEE 1219] et ISO/IEC 14764 (*Standard on Software Maintenance*) ont défini une quatrième catégorie à savoir la maintenance préventive.

- **Maintenance préventive** : elle est définie dans le but de prévenir les problèmes avant qu'ils n'apparaissent. Ce type peut être inclus dans la maintenance corrective, mais les communautés internationales des normes le classifient comme un type de maintenance à part.

2. Activités de maintenance : Les activités de maintenance sont considérées similaires à celles de développement du logiciel. Les mainteneurs doivent accomplir l'analyse, la conception, le code, le test et la documentation. Ils doivent ainsi répondre aux exigences de la même manière que celles du développement. Cependant, il y a une différence au niveau du processus. Le processus spécifique aux activités de maintenance du logiciel comporte les activités uniques, les activités de support et les activités de planification de maintenance.

- Activités uniques : Les mainteneurs doivent avoir des connaissances approfondies de la structure et du contenu du code. Contrairement au développement du logiciel, les mainteneurs doivent

accomplir l'analyse de l'incidence sur les coûts. Cette analyse est nécessaire pour déterminer les coûts relatifs à un changement. Une requête de modification doit être analysée puis traduite en terme logiciel. Le mainteneur identifie ainsi les composants affectés. Il peut y avoir plusieurs solutions potentielles, mais une recommandation devrait être préparée comme meilleure action.

- Activités de support : Les mainteneurs doivent également accomplir les activités de support telles que la gestion de configuration, la vérification, la validation, l'assurance-qualité, les révisions, l'audit, « operating help desk » et la gestion de la formation des utilisateurs. En ce qui concerne la gestion de configuration, il n'est pas suffisant de suivre simplement les requêtes de modification. Le produit logiciel et n'importe quel changement apporté à celui-ci doit être contrôlé. Ce contrôle est établi lorsque l'implantation du processus de gestion de configuration du logiciel est approuvé. Ce processus sera implanté en développant et en suivant un plan de gestion de configuration et des procédures opérationnelles. D'autre part, l'aspect qualité doit être pris en considération dans les processus de maintenance du logiciel. Parmi les attributs de qualité, on retrouve qu'il faut réduire la complexité du produit logiciel.

- Activité de planification de maintenance : Par cette activité, le plan de maintenance doit être préparé lors du développement d'un nouveau produit logiciel et doit spécifier « Comment » les utilisateurs demandent leurs modifications ainsi que définir la liste des problèmes évoqués (*Report Problems*). Le plan de maintenance devrait inclure :
  - l'objectif de la maintenance du logiciel,
  - l'ajustement du processus après la livraison,
  - la désignation du mainteneur,
  - l'estimation des coûts du cycle de vie.

3. Aspect d'organisation de maintenance : Généralement, l'équipe qui développe le produit logiciel n'est pas celle qui accomplit la maintenance. À cet effet, on retrouve plusieurs modes d'organisation de maintenance :

- Mainteneur : Il est souvent employé pour s'assurer que le système fonctionne bien et permet de satisfaire aux besoins de changements des utilisateurs [Pfleeger, 1998].

- Impartition : Elle est considérée comme étant le processus d'approvisionnement externe des biens matériels ou des services au lieu de les assurer par les propres moyens de l'organisation [Dorfman et Thayer, 1997].

- Structure organisationnelle : En se basant sur le fait qu'il y a presque autant de structures organisationnelles que d'organisations de maintenance du logiciel, Pigoski suggère de désigner un responsable dans le groupe de maintenance, indépendamment de la structure organisationnelle [Pigoski, 1997].

Nous avons remarqué que les sous-domaines de connaissance à savoir, le besoin pour la maintenance, les catégories de maintenance, les activités de maintenance et les aspects d'organisation de maintenance comportent des notions de base et des concepts relatifs à la maintenance. Ces concepts peuvent être des « principes opérationnels » associés avec « la configuration normale ». Probablement, c'est la raison pour laquelle Pigoski a placé ces sous-domaines dans le critère des «concepts fondamentaux de conception ».

2.2.2 Les critères et les spécifications

Dans ce critère, Pigoski inclut la section portant sur les mesures de maintenance du logiciel.

1. Mesures de maintenance du logiciel : Pigoski montre qu'une stratégie de maintenance est nécessaire pour minimiser les coûts et que les mesures de logiciels font partie de cette stratégie. Grady, quant à lui considère les mesures pour aboutir à l'amélioration du processus logiciel [Grady et Caswel, 1987]. Dans le même contexte, Takang évoque qu'une mesure est assumée pour l'évaluation, le contrôle, l'estimation, l'amélioration et la prédiction [Takang et Grubb, 1997].

- Établissement d'un programme de mesure : Il se base généralement sur le paradigme GQM « Goal, Question, Metric » de Basili [Stark, Kern et Vowell, 1994]. Ce paradigme montre qu'un programme de mesure consiste à : (i) identifier les objectifs organisationnels, (ii) définir les questions relatives aux objectifs et, (iii) sélectionner les mesures permettant de répondre aux questions.

La norme IEEE [ANSI/IEEE STD 1061] fournit une méthodologie pour établir les besoins de qualité, identifier, implanter, analyser et valider le processus et les mesures de qualité d'un produit logiciel. Cette méthodologie s'applique pour tout produit logiciel et dans toutes les phases du cycle de vie d'un produit logiciel y compris la maintenance.

- Mesures spécifiques : Ce sont des mesures propres à la maintenance du logiciel. Souvent les mesures générales du génie sont appliquées à la maintenance et le mainteneur doit déterminer celles qui sont les plus appropriées pour l'organisation. Stark suggère une liste de mesures pour la maintenance utilisée par le NASA [Stark et al., 1994]. Cette liste comporte :
  - taille du logiciel,
  - personnel du logiciel,
  - procédés des requêtes de maintenance,
  - procédés d'amélioration du logiciel,
  - programme de ressources des ordinateurs,
  - densité d'une erreur,
  - volatilité d'un logiciel,
  - durée d'un rapport de désaccords,
  - ratio break/fix,
  - fiabilité d'un logiciel,
  - complexité de conception,
  - distribution du type erreur.

En résumé, nous pouvons remarquer que les mesures de maintenance du logiciel permettent à l'ingénieur du système d'évaluer différents aspects de taille et des attributs de qualité. D'où leur considération dans le critère des « critères et spécifications ».

2.2.3 Les outils théoriques

Selon Pigoski, les connaissances de ce type de maintenance ne présentent aucun élément dans le présent critère des « outils théoriques ».

2.2.4 Les données quantitatives

On retrouve dans ce critère la section portant sur le coût de maintenance et l'estimation du coût de maintenance.

1. Coût de maintenance et estimation du coût de maintenance : Le coût est toujours capital lorsqu'il est question d'évolution et de maintenance du logiciel [Pfleeger, 1998]. Il est nécessaire de maîtriser les différentes catégories de maintenance pour adresser les coûts de la maintenance.

- Coût de maintenance : Pfleeger montre que la maintenance consomme plus que la moitié du coût total du cycle de vie des logiciels. La compréhension des différentes catégories de maintenance et les facteurs qui influencent la maintenabilité d'un système peuvent aider à cerner les coûts [Pfleeger, 1998].

- Estimation du coût de maintenance : plusieurs facteurs techniques et non-techniques affectent l'estimation du coût de la maintenance. Le modèle COCOMO de Boehm est conçu pour estimer les coûts d'un logiciel [Boehm, 1981]. L'estimation de maintenance est souvent basée sur les données empiriques et les expériences.

Dans cette division, nous avons remarqué l'existence de diverses approches pour l'estimation des coûts de maintenance. Celles-ci peuvent aider à déterminer les détails des systèmes à maintenir. Ceci explique pourquoi l'estimation du coût de maintenance a été classée dans le critère des « données quantitatives ».

2.2.5 Les considérations pratiques

Dans ce critère, Pigoski considère les trois éléments suivants :

1. Problèmes de maintenance du logiciel : Maintenir un système n'est pas une tâche facile. Puisque le système est déjà opérationnel, l'équipe de maintenance doit concilier entre le besoin de changement et le besoin de garder un système accessible aux utilisateurs. De plus, le fait de trouver des défauts dans le code que le mainteneur n'a pas développé est un défi pour le mainteneur. Il doit s'assurer des ressources fiables et d'une documentation mise à jour.

Les problèmes qui concernent la maintenance du logiciel étant principalement d'ordre technique et de gestion :

- Problèmes techniques : Ces problèmes concernent particulièrement la compréhension limitée, le test, l'analyse de l'impact et la maintenabilité.

Compréhension limitée : Une étude réalisée par Pfleeger montre que 40 à 60% de l'effort de maintenance est lié à la compréhension du code à modifier [Pfleeger, 1998]. Ceci est dû à un manque de documentation (comme mentionné au paragraphe 1. du chapitre 1). D'autre part, Swanson énonce que plus de la moitié des problèmes des mainteneurs sont dûs à un manque de capacité de compréhension des utilisateurs. Par exemple, il a constaté que les utilisateurs ne comprenant pas bien le fonctionnement d'un système, peuvent induire les mainteneurs en erreur avec des données incomplètes et trompeuses au moment de l'exécution d'une requête de modification [Swanson, 1989].

Test : Certains tests sur un logiciel peuvent être significatifs en terme de temps et d'argent. Ainsi, la détermination d'un sous-ensemble de tests pour s'assurer de la qualité des modifications apportées est un défi pour les mainteneurs.

Analyse de l'impact : Ce type d'analyse permet d'identifier tous les systèmes logiciels affectés par une requête de modification et permet l'estimation des ressources pour accomplir les changements. Elle est nécessaire pour minimiser le risque et elle est souvent accomplie après une demande de modification au niveau du processus de gestion de configuration.

Maintenabilité : Elle est définie par la norme ISO comme l'une des caractéristiques de qualité. Pigoski suggère d'incorporer la maintenabilité dans l'effort de développement d'un logiciel pour réduire les coûts du cycle de vie particulièrement dans les phases de spécification des besoins, de conception et de construction d'un logiciel. Elle peut être améliorée en définissant des normes pour le codage, la documentation et les outils de test dans la phase de développement du logiciel [Pigoski, 1997].

- Problèmes de gestion : Ce sont des problèmes liés à l'alignement avec les problèmes organisationnels, le personnel et les problèmes du processus.

Alignement avec les problèmes organisationnels : Le retour sur l'investissement n'est pas clair avec la maintenance. Ainsi, il y a une lutte constante pour obtenir les ressources [Dorfman et Thayer, 1997].

Personnel : le personnel de la maintenance ou le mainteneur souffre souvent d'un moral faible (souligné dans le paragraphe 1.1.2. Moral). Deklava fournit une liste des problèmes de personnel basée sur une enquête [Deklava, 1992].

Problèmes de processus : La maintenance nécessite plusieurs activités qui ne figurent pas dans le développement du logiciel comme par exemple le «*help desk support*» [Dorfman et Thayer, 1997].

2. Techniques pour la maintenance : Selon Pigoski, les techniques généralement reconnues dans la pratique sont les suivants : la compréhension du programme, la réingénierie, la rétro-conception et l'analyse de l'impact. Ces techniques sont introduites entre autre dans le document de [Dorfman et Thayer, 1997].

3. Ressources : Par ressources, Pigoski indique toutes les documentations disponibles sur la maintenance du logiciel.

Les problèmes de maintenance du logiciel, les techniques pour la maintenance et les ressources sont des éléments basés sur des expériences. D'où leurs inclusions dans le critère des « considérations pratiques ».

2.2.6 Les instruments de conception

1. Processus de maintenance : La section portant sur le processus de maintenance du logiciel est bien documentée. Le modèle CMM (*Capability Maturity Model*) du SEI (*Software Engineering Institute*) fournit un moyen de mesurer les niveaux de maturité. L'intérêt est qu'il présente une corrélation directe entre les niveaux de maturité et les coûts économisés.

- Modèles du processus de maintenance : Les modèles de processus de maintenance sont inspirés des normes IEEE [IEEE1219] et ISO/IEC 14764.

Dans cette dernière division des modèles de processus de maintenance du logiciel, il a été constaté que de tels modèles concernent réellement le « Comment » procéder à la maintenance. Pour ce faire, ils sont inclus dans le critère des « instruments de conception ».

2.3 CONCLUSION

Nous avons analysé dans ce chapitre une démarche de décomposition du domaine de maintenance déjà formulée par Pigoski, une démarche qui a pour but de vérifier l'application des critères de Vincenti de l'ingénierie et de retenir les connaissances de maintenance les plus près de la maturité. Mais un ou deux critères ne sont pas bien définis pour les appliquer directement dans le domaine des connaissances de maintenance, bien qu'ils aient été utilisés pour l'essai de classification selon les critères de Vincenti. En effet, nous avons remarqué qu'il y a un manque d'éléments dans le critère des « outils théoriques » et que les connaissances de maintenance présentent des faiblesses. D'autre part, il n'est pas évident de combler ce vide même dans le cas où Ho considère qu'il y a plutôt un vide dans le critère des « données quantitatives ». Les deux classifications (Pigoski et Ho) ne sont pas complémentaires l'une de l'autre car chacun procède selon sa perception et les deux n'ont pas fourni

assez ou pas du tout d'explication détaillée ou d'arguments solides sur le regroupement des éléments dans un critère donné de la classification de Vincenti.

En résumé, nous constatons que dans l'application des critères de Vincenti, certains éléments de connaissance sont clairement discernables, d'autres ne le sont pas. Du point de vue de l'ingénierie, les connaissances de maintenance ne sont pas suffisantes et assez matures.

L'analyse faite par Pigoski, selon les critères de Vincenti, présente ainsi une décomposition du domaine des connaissances de maintenance en sous-domaines dont la signification n'est pas immédiatement évidente. Nous retiendrons son analyse et procéderons par analogie dans les deux chapitres suivants portant sur les travaux de Pigoski, pour le premier, et la norme ISO/IEC 14764 pour le deuxième.

CHAPITRE III

ANALYSE DU MODÈLE DE PIGOSKI SELON LES CRITÈRES DE VINCENTI

Dans ce chapitre, nous décrivons le texte de Pigoski « Practical Software Maintenance : *Best Practices for Managing your Software Investment*, 1997 » tout en présentant l'intérêt d'avoir choisi ce document. Par la suite, nous analyserons ce texte selon la description de Vincenti des critères des connaissances de l'ingénierie.

3.1 MODÈLE DE MAINTENANCE DE PIGOSKI

Dans son livre «*Practical Software Maintenance : Best Practices for Managing your Software Investment* », Thomas Pigoski introduit un aspect pratique des modèles de maintenance du logiciel. Contrairement à d'autres auteurs qui ont discuté de ce sujet, Pigoski considère la maintenance des logiciels dans un contexte moderne des modèles de processus des logiciels; ceci présente donc un intérêt pour la discipline du génie logiciel.

L'objectif de l'auteur de ce livre est de partager aussi bien les pratiques que les expériences de la maintenance des logiciels avec les professionnels de la maintenance, c'est-à-dire, ceux qui s'intéressent à connaître ce qui est important dans le domaine de maintenance. D'autre part, ce livre permet de combler le vide qui existe entre la théorie et la pratique, de fournir des instructions précieuses et d'offrir des suggestions sur l'amélioration de l'état de maintenance du logiciel.

3.2 ANALYSE DU MODÈLE DE MAINTENANCE DE PIGOSKI

Dans cette section, nous allons analyser le modèle de maintenance de Pigoski en procédant de façon analogue à ce qu'il a réalisé dans le projet SWEBOK. Le tableau 3.4 résume notre suggestion de décomposition des connaissances de maintenance qui se trouvent dans le livre de Pigoski, selon la description de Vincenti des critères de connaissances de l'ingénierie.

Tableau 3.3 : Récapitulatif du texte de Pigoski sur la maintenance selon les critères de Vincenti

Texte de Pigoski – Practical Software Maintenance
1. LES CONCEPTS FONDAMENTAUX DE CONCEPTION
Maintenance des logiciels Activité pré-livraison de maintenance du logiciel Planning : Concept de maintenance, plan de maintenance et ressources Organisations de maintenance du logiciel
2. LES CRITÈRES ET LES SPÉCIFICATIONS
Mesures de maintenance du logiciel
3. LES OUTILS THÉORIQUES
4. LES DONNÉES QUANTITATIVES
Expériences de mesures de maintenance des logiciels
5. LES CONSIDÉRATIONS PRATIQUES
Transition Outils et environnement Maintenabilité Gestion de maintenance des logiciels Éducation et formation Impact de la technologie orientée objet sur la maintenance
6. LES INSTRUMENTS DE CONCEPTION
Processus de maintenance de l'ISO/IEC 12207 : Implantation du processus Analyse du problème et de modifications Implantation de modifications Revue/acceptation de maintenance Migration Retraite du logic iel

3.2.1 Les concepts fondamentaux de conception

Le critère des « concepts fondamentaux de conception », nous l'avons dit dans les chapitres précédents, consiste à des principes opérationnels associés à des configurations normales d'un système à concevoir, c'est-à-dire elle consiste en deux aspects permettant de définir une approche de conception. Cette approche nous aide à mieux comprendre les concepts et les notions de base liés à un domaine de l'ingénierie, particulièrement celui de la maintenance. Pour cela, nous incluons dans le présent critère les éléments suivants :

1. Maintenance des logiciels :

Nous avons parlé, au paragraphe 2.1.1 du chapitre 2, de la nécessité de la fonction de maintenance pour garder utilisable un système logiciel. Il faut ajouter œci : dans la pratique, le financement pour effectuer des recherches sur cette fonction est inexistant [Pigoski, 1997]. Les communautés des académiciens ainsi que les praticiens publient très peu sur ce sujet et, relativement, peu de livres portent spécifiquement sur la maintenance du logiciel. Ce manque d'information induit à un manque de compréhension de cette fonction [Arthur, 1988].

Dans son livre, Pigoski s'est clairement exprimé sur ce qui se produit lors de la maintenance des logiciels et les types de changements qui peuvent être effectués. Il a indiqué que la maintenance permet de corriger les erreurs, de prévoir les problèmes avant qu'ils n'apparaissent et d'améliorer la performance d'un système logiciel. Il a précisé également les différents travaux réalisés pour classifier les types de maintenance. La classification de Swanson, qui est considérée comme le premier travail dans ce sens et qui a été repris par plusieurs auteurs, regroupe la fonction maintenance en trois catégories selon les causes constatées : maintenance corrective, maintenance adaptative et maintenance perfective.

Tout d'abord, la **maintenance corrective** représente les changements nécessités par des erreurs effectives (erreurs produites à un système). Ensuite la **maintenance adaptative** constitue n'importe quel effort qui est initié comme un résultat de changements dans un environnement où un produit logiciel doit opérer. Enfin, le troisième type de maintenance est **la maintenance perfective** représentant tous les changements, les insertions, les suppressions, les modifications, les extensions et les améliorations réalisés à un système pour répondre aux besoins des utilisateurs.

D'ailleurs, Pigoski propose de garder deux groupes de maintenance : la maintenance corrective et les améliorations (*"enhancements"*) qui sont la fusion de la maintenance perfective et la maintenance adaptative.

2. Activités pré-livraison de maintenance du logiciel :

Cette section du livre de Pigoski comporte plusieurs études de cas pour renforcer les concepts et pour fournir des expériences pratiques additionnelles. Cependant, selon la définition des critères de Vincenti, les activités de maintenance du logiciel devront être plutôt incluses dans le critère des

« instruments de conception ». Malgré cet aspect pratique, elles nous renseignent assez bien sur les activités qu'un mainteneur doit entreprendre et les notions de base qu'il doit posséder. À cet effet, nous la plaçons dans le critère des « concepts fondamentaux de conception ».

Pigoski propose aux mainteneurs de sélectionner les approches qui sont considérées appropriées à leur organisation de maintenance et les inclure dans une implantation de ISO/IEC 12207. Il précise que les activités de maintenance du logiciel changent à travers les étapes de livraison. Certaines de ces activités devront être accomplies durant l'étape pré-livraison, que ce soit par les développeurs ou par une équipe séparée de mainteneurs. Dans le contexte de ISO/IEC 12207, l'activité pré-livraison est accomplie durant l'activité d'implantation du processus logiciel. Celle de Post-livraison commence avec l'implantation du processus et exécute toutes les étapes relatives à l'abandon du produit logiciel. Si l'activité de maintenance est effectuée par le développeur, il est nécessaire que ce dernier planifie et fournisse les coûts effectifs de maintenance. Pigoski souligne que, dans la pratique, les développeurs ne se concentrent pas sur ce genre de problèmes tels que le support et les coûts du cycle de vie d'un système logiciel. D'autre part, l'implication du mainteneur durant l'étape pré-livraison permet de réduire les coûts futurs du cycle de vie. Le mainteneur devrait concilier entre les différentes visions des clients, des utilisateurs et des développeurs. Ces derniers, s'intéressent généralement à avoir un nouveau système de façon rapide et économique. Le mainteneur cherche plutôt à s'assurer que le nouveau système soit supportable et puisse être maintenu à un coût effectif. À cet effet, Pigoski énonce que la meilleure approche est celle de l'organisation logistique. Celle-ci permet de :

- développer une stratégie de support pour la maintenance,
- désigner le mainteneur,
- investiguer des concepts alternatifs de support,
- aider à définir la portée de maintenance,
- aider à définir le système de support de maintenance (matériel et logiciel).

3. Planning : concept de maintenance, plan de maintenance et ressources :

L'étude des sections du livre de Pigoski (concept de maintenance, plan de maintenance et ressources) nous aide à mieux cerner l'effort que le mainteneur doit fournir pour avoir un planning adéquat et approprié de maintenance. Le mainteneur devra disposer des connaissances assez complètes et définitives des conditions de planification. C'est ce point de vue sur lequel Pigoski attire notre attention. Malgré la présence des exemples pratiques dans les différentes sections, il ne s'agit pas de mettre l'accent sur les détails des considérations pratiques, mais plutôt sur les notions de base que le

mainteneur doit acquérir à propos du planning de maintenance. À cet effet, nous considérons le planning dans le critère des « concepts fondamentaux de conception ».

Selon Pigoski, il n'est pas facile d'attribuer les responsabilités pour accomplir la maintenance du logiciel. En effet, les développeurs cherchent souvent à avoir un système logiciel adéquat dans un temps limité et un budget bien déterminé. Le mainteneur, quant à lui, se concentre sur le support du système. Ainsi, il n'est pas évident de concilier entre les objectifs principaux des développeurs et ceux des mainteneurs et même des clients. À cet effet, Pigoski nous propose de déterminer un planning pour alléger ce conflit. Ce planning de maintenance sera préparé pour les ressources matérielles et humaines nécessaires pour maintenir les systèmes livrés et réduire ainsi les coûts du cycle de vie. Un planning adéquat doit être composé des éléments suivants : concept de maintenance, plan de maintenance et ressources.

Concernant le concept de maintenance, il doit être développé au début de l'effort de développement par l'utilisateur avec l'assistance du mainteneur. Ce concept doit souligner :

- la portée de maintenance du logiciel,
- l'adaptation au processus après-livraison,
- la désignation de celui qui réalise la maintenance,
- l'estimation des coûts du cycle de vie.

Quant au plan de maintenance, il doit être préparé durant le développement du logiciel et doit comporter les spécifications nécessaires pour réaliser les modifications du système. Il doit également définir l'assurance-qualité et les activités de tests.

Enfin, Pigoski classifie les ressources selon trois sortes : les ressources du personnel, les ressources d'environnement et les ressources financières.

- Les ressources du personnel

Un des problèmes majeurs dans le planning de maintenance du logiciel est le problème de planification des besoins en ressources de maintenance. Les exigences du personnel constituent un facteur de coût considérable, qui est le plus difficile à estimer. Pigoski énonce que la détermination des ressources du personnel peut s'effectuer selon deux approches. La première consiste à utiliser les modèles paramétriques tels que le modèle COCOMO [Boehm, 1981] et les modèles GECOMO, REVIC, SLIM,

CHECKPOINT et SOFTCOST [Jones, 1994]. La seconde se base sur l'expérience, c'est-à-dire sur les données empiriques et historiques qui permettent de déterminer les besoins en maintenance. Pigoski évoque également la notion des ressources de personnel à l'égard de ce qui se trouve en pratique. Il s'appuie sur une étude de cas pour illustrer la façon de déterminer les exigences des ressources du personnel pour la maintenance.

- Les ressources d'environnement

Dans cette partie, Pigoski met l'accent sur l'établissement d'un plan pour l'environnement de maintenance. Ceci devrait être pris en compte au moment où les ressources sont allouées et les budgets sont attribués pour le développement et la maintenance du système logiciel.

- Les ressources financières

Ce troisième aspect de ressources est nécessaire pour fournir un support effectif de maintenance. Pigoski évoque le fait que le budget de maintenance doit être déterminé dés le départ pour avoir un planning de maintenance approprié.

4. Organisations de maintenance du logiciel :

Dans cette section, Pigoski décrit les fonctions de maintenance dans un contexte organisationnel et discute l'arrangement des différentes organisations de maintenance tout en présentant un échantillon de directives de maintenance. Il parle du développement d'une organisation pour une organisation de maintenance du logiciel et des facteurs dépendants. Il traite ces facteurs du point de vue politique et non pas du point de vue technique, c'est-à-dire qu'il décrit la manière que le mainteneur peut suivre pour conduire une organisation de maintenance. Dans ce sens, nous considérons la directive d'organisation de maintenance comme un énoncé décrivant des idées fondamentales et nous traitons ainsi cette section dans le premier critère des connaissances de l'ingénierie de Vincenti.

Pigoski énonce que les rôles, les responsabilités et les fonctions du mainteneur doivent être clairement définis. Pour cela, il est nécessaire d'avoir des documents de haut niveau (*'higher-level'),* des directives ou une politique de maintenance du logiciel permettant de clarifier et de raffiner les fonctions de l'organisation de maintenance. Ces directives doivent garder trace de :

- toutes les responsabilités, les autorités et les fonctions de l'organisation de maintenance,
- les opérations de l'organisation de maintenance du logiciel,
- le besoin et la justification pour les modifications,
- les responsabilités pour la réalisation des modifications,
- les procédures et le contrôle de modifications,
- le processus et les procédures pour contrôler les modifications du logiciel.

3.2.2 Les critères et les spécifications

Nous avons vu au paragraphe 1.6 du chapitre 1 que le critère des « critères et des spécifications » est un critère permettant au concepteur de traduire les buts qualitatifs d'un système (le général) en des buts quantitatifs traduits dans des termes techniques concrets (le spécifique). Cette traduction met en relief l'aspect d'une méthode permettant d'obtenir des résultats concrets, précis et communicables sans ambiguïté. En ce sens, nous considérons cette méthode comme étant le procédé qui permet de traduire ce qui est mesuré indirectement (le qualitatif), selon son rapport avec celui qui est mesuré directement (le quantitatif). Ainsi, nous traiterons de la mesure des logiciels dans ce critère des « critères et des spécifications ».

1. Mesures de maintenance du logiciel :

Dans cette section, Pigoski introduit, pour des raisons de cohérence théorique, les concepts de mesures, et mentionne les difficultés survenues lors de l'établissement d'un programme de mesures et de la détermination du choix des mesures à utiliser. Il énonce également que la définition des objectifs organisationnels constitue la première étape dans l'établissement d'un programme de mesures de logiciels et s'appuie sur des études de cas pour donner des informations spécifiques et pratiques à propos des mesures.

Selon Grady, une mesure de logiciel est une façon pour mesurer certains attributs d'un processus logiciel tels que par exemple la taille, le coût par défaut, la communication, la difficulté et l'environnement [Grady et Caswell, 1987]. L'application des mesures induit à une meilleure compréhension et prédictibilité croissante du processus. Cependant, il est nécessaire d'avoir un processus documenté pour profiter de l'utilité des mesures.

Card énonce que lors de l'évaluation d'un processus logiciel, la majorité des organisations mettent l'accent sur la qualité du produit à travers des inspections et des révisions. Ces dernières, se font manuellement et renforcent simplement les normes [Card et Glass, 1990]. Pour cela, ces pratiques

n'assurent pas une qualité des logiciels. Il suffit donc d'utiliser des méthodes numériques pour mesurer la qualité.

Selon la norme IEEE [ANSI/IEEE STD 1061], une mesure est synonyme avec une « mesure de qualité d'un logiciel » et elle est définie comme étant une fonction avec des entrées et des sorties. Les mesures de qualité d'un logiciel ont comme entrées des données logiciels et comme sorties une valeur numérique unique. Les sorties sont interprétées comme le degré auquel un logiciel possède un attribut donné qui affecte sa qualité.

Selon la norme ISO/IEC 9126, une mesure de qualité des logiciels est définie comme étant une échelle quantitative et une méthode qui peut être utilisée pour déterminer la valeur d'une caractéristique d'un produit logiciel donné.

Pigoski quant à lui, considère que dans la majorité des organisations, les mesures représentent un moyen permettant de mesurer certains aspects du logiciel qui sont liés directement à la qualité. De plus, il montre qu'il est presque impossible d'établir un programme de mesures sans avoir des objectifs concis.

Différentes approches ont eu un succès d'implantation d'un programme de mesure :

1. L'approche de Grady [Grady, 1992] est la suivante :

- définir l'objectif pour un programme,
- assigner les responsabilités,
- faire les recherches,
- définir les métriques initiales de collection,
- avoir les outils pour la collection et l'analyse des données,
- établir une classe de formation sur les métriques logiciels,
- encourager l'échange des idées,
- créer une base de données des mesures,
- établir un mécanisme pour changer la norme selon une manière appropriée.

2. L'approche de Card [Card et Glass, 1990] consiste à :

- définir l'objectif de mesure (exemple le logiciel),
- identifier les caractéristiques ou les attributs à mesurer (exemple la qualité),
- spécifier l'objectif ou l'utilisation des résultats de mesure,
- collecter les données,
- vérifier et modifier le modèle basé sur l'analyse et l'application avec les données collectées.

3. L'approche de Basili [Basili, 1985] fondée sur l'approche GQM (*Goal, Question, Metric*) consiste à :

- identifier les objectifs organisationnels,
- définir les questions relatives aux objectifs,
- sélectionner les mesures qui répondent aux questions.

En 1992, le DoD (*Departement of Defense*) des États-Unis avait proposé un projet de mesure des processus logiciels de SEI afin de développer le matériel et le guide d'un ensemble de mesures utiles pour l'acquisition, le développement et les programmes de support de logiciel du DoD. L'objectif étant de produire des mesures permettant de servir comme une base de collection des données consistantes et compréhensibles par le DoD. Afin d'atteindre cet objectif, ils se sont basés sur les mesures de gestion concernant la taille, l'effort, le programme et la qualité.

La taille : Toutes les organisations de maintenance doivent connaître la taille de leur portefeuille d'application et la taille de chaque produit (ou application) si la taille est déterminée par le nombre de ligne de code ou les points de fonction. L'organisation de maintenance doit suivre le changement de la taille pour chaque produit et utilise la taille pour déterminer le trafic de changement annuel.

L'effort : Les gestionnaires ont besoin des mesures bien définies et compréhensibles universellement pour l'estimation du projet logiciel, la planification et le suivi. Les mainteneurs doivent suivre l'effort en terme d'heures du personnel pour les activités de maintenance.

Le programme : Le cadre pour la définition des programmes vise deux aspects différents de mesure de programme. Le premier aspect concerne les données du projet et les livrables. Le second concerne les mesures de progression. Le suivi des données et des livrables permet d'avoir une vue d'un niveau macro d'un programme de projet.

La qualité : Le rapport de SEI fournit un cadre pour la découverte, le rapport et la mesure des problèmes logiciels. Ce rapport indique les défauts et les problèmes rapportés dans différents formats de données par différents défauts trouvés dans les activités. L'objectif est de fournir des méthodes qui peuvent aider à obtenir des mesures claires et consistantes de qualité basées sur une variété de « *Problem Reports* » de logiciels et des données dérivées par l'analyse et des actions correctives.

3.2.3 Les outils théoriques

En se basant sur la décomposition des domaines de maintenance réalisée par Pigoski dans le projet SWEBOK, nous pouvons constater, par analogie, qu'il y a un manque d'éléments dans le critère des « outils théoriques ».

3.2.4 Les données quantitatives

Nous avons mentionné dans le chapitre 1 au paragraphe 1.6.1 que les données quantitatives sont des données obtenues de façon empirique conformément aux propriétés physiques d'un système et ce pour aider à déterminer les détails du système à maintenir. Ces données sont représentées typiquement dans des tableaux ou des graphes et subdivisées en deux sortes de connaissances. La première est descriptive permettant de spécifier « Comment sont les choses? ». La seconde est prescriptive permettant de préciser « Comment devraient être les choses? » pour atteindre l'objectif final. Ainsi, nous pouvons constater que les « données quantitatives » sont obtenues en se basant sur des expériences ou des observations. En d'autres termes, avec ces données, on peut décrire les opérations de détection ou de mesure qu'il faut effectuer pour atteindre à définir les activités de maintenance. Pour cela, nous incluons dans le présent critère la section qui porte sur les expériences de mesures de maintenance du logiciel.

1. Expériences de mesure de maintenance des logiciels :

Pigoski nous illustre deux études de cas. La première consiste à examiner l'évolution des programmes de mesure et la seconde consiste à analyser des mesures de maintenance collectées dans une période de six ans au sein d'une organisation de maintenance des logiciels aux États-Unis. Il nous montre qu'un programme de mesure permet de fournir un logiciel de qualité aux utilisateurs et que pour chaque requête de modification ou des actions correctives, l'organisation de maintenance peut déterminer la cause de l'erreur. Mais dans la réalité, ces requêtes sont attribuées au développeur sous forme de défauts en latence.

3.2.5 Les considérations pratiques

Le critère des « considérations pratiques » repose sur des expériences ou des conventions dans la pratique. Contrairement, aux « données quantitatives » qui sont précises et formulées à partir des recherches délibérées, les propriétés des « considérations pratiques » sont susceptibles de changer.

Cependant, pour maintenir un système de qualité, les mainteneurs auront besoin des considérations dérivées des expériences dans la pratique. Ces considérations ne figurent pas dans la théorie mais plutôt apprises dans la vie active. À cet effet, nous traiterons dans cette catégorie les éléments suivants :

1. Transition :

Dans cette section, Pigoski nous montre les différents problèmes qui peuvent surgir lors d'une transition, les solutions possibles à ces problèmes et le modèle de transition nécessaire pour les mainteneurs. Il énonce qu'une organisation de maintenance doit être impliquée dès le départ dans l'effort de développement d'un système et que cette condition n'est pas certaine lorsqu'il s'agit d'impartition.

La transition de logiciel est définie comme étant une séquence d'actions contrôlées et coordonnées à travers lesquelles le produit logiciel passe de l'organisation qui exécute le développement initial du logiciel vers l'organisation qui exécute le support après sa livraison. Le mainteneur devrait énoncé lors du développement les responsabilités à assumer pour la maintenance. Ces responsabilités concernent, particulièrement, la transition de développement vers la maintenance. Ainsi, il est nécessaire d'établir un plan de transition. Celui-ci, doit couvrir tous les éléments de transition du logiciel (la transition de développement du logiciel et la transition du mainteneur) et doit utiliser certains éléments du plan de maintenance. Cependant, il est séparé et distinct d'un plan de maintenance.

Un plan de maintenance souligne tous les problèmes de haut niveau, il vise le transfert réel des responsabilités à partir du développeur vers le mainteneur et inclue tous les plans d'action. Il contient toutes les formalités de correspondance selon l'utilisation d'une maintenance convenable et les recherches nécessaires pour effectuer la transition.

Les problèmes majeurs qui peuvent apparaître lors d'une transition d'un système logiciel du développeur au mainteneur sont :

- le transfert des connaissances,
- la documentation,
- la communication avec les développeurs,
- la communication avec les utilisateurs,
- le processus formel,
- la formation,
- la rapidité de transition,

- la détermination de celui qui va accomplir la transition,
- le coût de transition.

2. Outils et environnement :

Dans cette section, Pigoski fournit l'environnement des outils CASE nécessaires pour accomplir les activités de maintenance, les techniques à suivre, la détermination d'un environnement CASE intégré ainsi que les spécifications du concept COSEE « Common integrated software engineering environment » qui permet de réduire les coûts du cycle de vie d'un système à maintenir. Ce concept COSEE fournit un environnement du génie logiciel ouvert et flexible qui est apte à évoluer.

3. Maintenabilité :

Pigoski traite de la maintenabilité des systèmes logiciels à partir d'une perspective du mainteneur et la considère comme étant un attribut de qualité. Plusieurs définitions ont été attribuées au terme maintenabilité. En effet, elle signifie :

- la facilité selon laquelle un système logiciel peut être modifié pour corriger les erreurs, améliorer la performance ou s'adapter à un environnement modifié. Ou encore, la facilité selon laquelle un système matériel peut être maintenu ou restauré afin d'accomplir ses fonctions requises [IEEE 610.12].
- la facilité selon laquelle un système logiciel peut être corrigé lors de l'apparition des erreurs et peut être déployé ou rétréci afin de répondre à des nouveaux besoins [Martin et McClure, 1983].
- les changements qui sont limités pour localiser les domaines (ou les modules) du système et qui sont faciles à réaliser [Card et Glass, 1990].
- un ensemble d'attributs portant sur l'effort nécessaire à réaliser les modifications spécifiées [ISO 8402].
- la facilité de garder un logiciel en opération après avoir réalisé des modifications au code. Cette facilité est énoncée en terme de temps et efforts requis pour accomplir ces modifications [Von Mayrhauser, 1990].

Afin de mieux déterminer les coûts de maintenance des logiciels, les mainteneurs doivent mesurer la maintenabilité des systèmes livrés et utiliser la maintenabilité des systèmes comme un facteur de détermination des coûts. Les caractéristiques de maintenabilité doivent être incorporées dans l'effort de développement du logiciel afin de réduire les coûts de maintenance du logiciel. Les facteurs les plus important qui affectent la maintenabilité sont : le processus de développement, la documentation et la compréhension du programme.

4. Gestion de maintenance des logiciels :

Il y a plusieurs problèmes de gestion qui sont liés à la maintenance des logiciels. Ces problèmes sont dus essentiellement à :

- un manque de personnel de maintenance, particulièrement les mainteneurs expérimentés,
- un manque de formation propre aux mainteneurs,
- un changement élevé provoquant un manque d'expertise,
- des difficultés de mesure de contribution,
- un moral faible dû à un manque de reconnaissance et de respect,
- une grande quantité de travaux en attente ou en retard (' *backlog')*,
- un changement des priorités,
- un manque d'une méthodologie de maintenance, de normes, de procédures et d'outils.

Pigoski énonce qu'il faut prendre en considération que la maintenance fait partie du génie logiciel et que pour réussir, le génie logiciel (incluant la maintenance) doit adresser le processus de l'ingénierie comme les industriels adressent leurs processus. De plus, les problèmes particuliers des praticiens de maintenance des logiciels doivent être soulignés tels que la vision des utilisateurs et des consommateurs vis à vis des praticiens de maintenance.

5. Éducation et formation :

Pigoski énonce que malgré l'importance de l'éducation et la formation de l'ingénieur du logiciel en ce qui concerne la maintenance, dans la pratique celle-ci n'est pas bien complète. En effet, ni les gens de l'industrie, ni les académiciens n'ont suffisamment étudié ce problème. L'industrie ne s'intéresse pas à l'éducation des concepts de maintenance à ses nouveaux employés recrutés; l'industrie dépend étroitement du milieu de l'éducation. À cet effet, Pigoski nous fournit des suggestions sur « Comment » et «Quoi » inclure dans un programme d'éducation de maintenance ainsi que les expériences pratiques dans l'enseignement de maintenance à un niveau universitaire.

L'éducation devrait fournir les concepts, l'information et les connaissances générales utiles pour le mainteneur pour accomplir sa fonction de maintenance. D'autre part, la formation est utile pour :

- se familiariser avec le nouveau système ou le produit logiciel,
- introduire le processus du mainteneur,
- étudier l'utilisation des outils spécifiques de maintenance.

6. Impact de la technologie orientée objet sur la maintenance :

Dans cette section, Pigoski fournit les concepts de la technologie orientée objet, discute de la vision des praticiens de maintenance sur cette technologie et décrit comment cette technologie peut influencer la maintenance des logiciels. Ainsi, la technologie orientée objet a un impact sur les mainteneurs et les développeurs des systèmes logiciels. Malgré les avantages offerts par cette technologie, en introduisant le support de nouvelles caractéristiques comme l'encapsulation, le mécanisme d'héritage et le polymorphisme (surcharge sémantique), ces dernières génèrent de nouveaux problèmes lors de la maintenance des applications développées à l'aide des outils orienté objets comme le problème d'analyse des impacts sur les composants de l'application après une modification.

3.2.6 Les instruments de conception

En plus des « considérations pratiques » et des « données quantitatives » requises pour leurs tâches, les ingénieurs ont besoin de connaître « comment accomplir ces tâches? », c'est-à-dire, des instruments de processus : procédures, façon de penser (incluant la pensée visuelle) et aptitude de jugement. Ces connaissances relèvent des expériences pratiques. Nous considérons le processus de maintenance des logiciels de la norme ISO/IEC 12207 comme un moyen permettant d'apporter des informations sur certains aspects de maintenance.

1. Processus de maintenance :

Dans cette section, Pigoski examine le processus de maintenance selon la norme ISO/IEC 12207 et fournit des interprétations sur la façon d'implanter ce processus dans l'objectif de définir, contrôler et améliorer l'effort de maintenance. Il considère qu'un processus de maintenance du logiciel de qualité permet de renforcer l'idée que la maintenance du logiciel fait partie intégrale d'une discipline complète du génie logiciel.

Les objectifs du processus de maintenance de ISO/IEC 12207 consistent à résoudre les problèmes, adapter le logiciel à des nouveaux environnements et implanter les améliorations tout en gardant l'intégrité du système. Ce processus commence quand il y a un besoin de maintenir un produit logiciel et se termine lorsque ce produit devient finalement en retraite. La figure suivante énumère les activités de ce processus et indique l'utilisation interne, le processus invoqué ainsi que les extrants associés à chacune de ces activités.

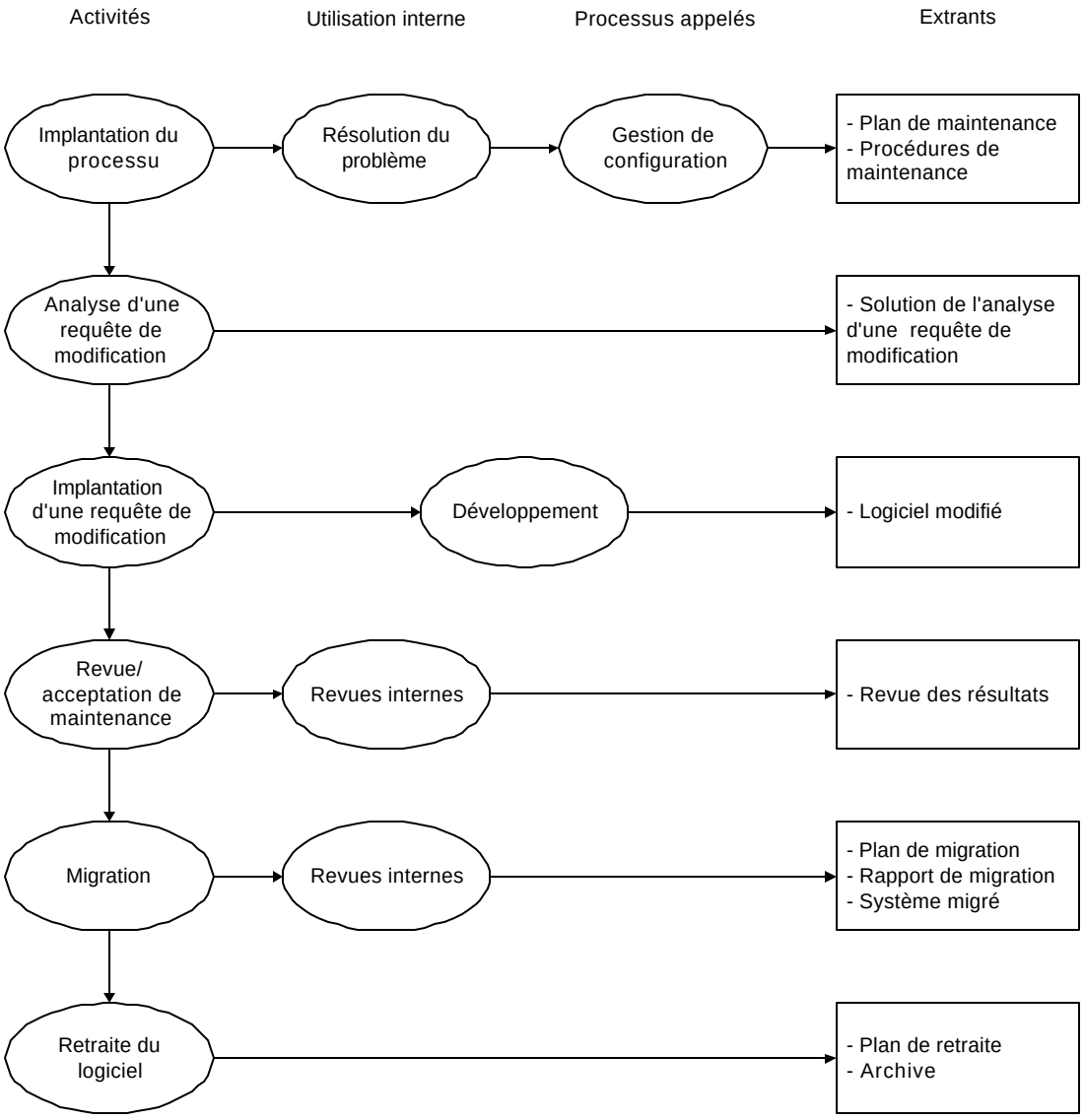

Figure 3.4 : Processus de maintenance de ISO/IEC 12207 [Pigoski, 1997]

Les points subséquents montreront les principales activités du processus de maintenance telles que décrites dans la norme ISO/IEC 12207 :

- **Implantation du processus** : durant cette activité, le mainteneur établit le plan de maintenance et les interfaces organisationnelles nécessaires.
- **Analyse de modification et de problèmes** : durant cette activité, le mainteneur évalue les requêtes de modification pour comprendre le problème, développer une solution et obtenir l'accord pour l'implantation d'une solution spécifique.

- **Implantation des modifications** : cette activité contient toutes les actions de conception, d'implantation, de test et de livraison utiles pour résoudre les requêtes de maintenance.
- **Revue/acceptation de maintenance** : cette activité est utilisée pour s'assurer qu'un produit logiciel est rectifié. Cette activité invoque les processus d'assurance-qualité, de vérification et de validation.
- **Migration de logiciel** : cette activité permet le mouvement du produit logiciel à un nouveau processeur. En d'autres termes, ce type de changement est défini comme étant la migration du logiciel d'un ancien à un nouvel environnement opérationnel.
- **Retraite de logiciel** : cette action se réalise lorsqu'un produit logiciel n'est plus utile et doit être retiré.

3.3 CONCLUSION

Ce chapitre constitue le résultat de l'analyse du modèle de maintenance de Pigoski. Ce résultat comprend la description des différentes sections du livre et l'explication de l'attribution de ces sections au niveau des catégories de Vincenti. Par analogie avec le chapitre précédent, nous avons remarqué que le domaine des connaissances de maintenance du logiciel n'est pas assez mature et présente des insuffisances dans le critère des « outils théoriques ».

Ce sujet a été traité dans le contexte d'une comparaison entre SWEBOK, le contenu de ce texte de Pigoski et la norme ISO/IEC 14764 sur la maintenance. Le chapitre suivant présentera, de la même façon, l'analyse de la norme ISO/IEC 14764 sur la maintenance.

CHAPITRE IV

ANALYSE DU MODÈLE DE ISO/IEC 14764 SELON LES CRITÈRES DE VINCENTI

Ce chapitre traite le modèle des connaissances de maintenance du logiciel tel que décrit dans la norme ISO/IEC 14764. Notre étude consiste à apporter des suggestions d'amélioration dans ce modèle. Tout d'abord, nous présenterons cette norme internationale fondée sur la description de l'ISO/IEC JTC1 (*Information Technology*, *Subcommittee SC7*, *Software Engineering*). Ensuite, nous aborderons l'analyse du modèle de l'ISO/IEC 14764 selon les critères de Vincenti pour faire ressortir les forces et les faiblesses de ce modèle.

4.1 DÉVELOPPEMENT DE LA NORME ISO/IEC 14764

L'ISO « *International Organization for Standardization* » est une fédération d'organismes nationaux de normalisation (corpus membres de l'ISO). Le travail de normalisation est exécuté par les comités techniques de l'ISO, composés d'experts nommés par les pays participants à ISO. Les organisations internationales, gouvernementales et non-gouvernementales, avec un statut officiel de liaison avec ISO, prennent part également dans ce travail de normalisation. L'ISO collabore étroitement avec l'IEC « *International Electrotechnical Commission* » sur toutes les matières de standardisation électrotechnique. D'origine relativement récent (1947), le terme « ISO » est dérivé du mot grec, *ISOS*, et veut dire littéralement « égale ». ISO est utilisée pour solliciter les idées des « experts mondiaux» et pour aboutir à une normalisation mondiale.

Les versions intérimaires des textes préparés par les comités techniques sont distribuées ensuite aux corpus membres pour le vote. La publication comme norme internationale nécessite l'approbation d'au moins 75% des corpus membres participant à un vote (ISO/IEC 14764 JTC 1/SC 7 N°2043). Cette organisation internationale de normes favorise essentiellement le développement de la normalisation mondiale des procédés, outils et technologies qui supportent l'ingénierie des produits et des systèmes logiciels.

4.1.1 Portée de la norme

La norme internationale ISO/IEC JTC1/SC7 N2043 – FDIS 14764 traite en détail de la gestion de processus de maintenance du logiciel comme décrite dans la norme ISO/IEC 12207. En plus, elle établit les définitions des différents types de maintenance et fournit une directive s'appliquant au planning, l'exécution, le contrôle, la révision et l'évaluation ainsi que l'achèvement du processus de maintenance.

Cette norme internationale fournit un cadre dans lequel des plans généraux et spécifiques de maintenance du logiciel peuvent être exécutés, évalués et adaptés à la portée d'un produit logiciel donné. Elle précise les terminologies et les procédés pour la maintenance du logiciel (outils, techniques et méthodes). Elle fournit également une directive pour la maintenance du logiciel et prescrit les activités de maintenance du logiciel ainsi que les besoins de planning de maintenance. Elle ne fournit ni opérations de logiciel ni fonctions opérationnelles comme par exemple : copie de secours, reprise, administration du système qui sont accomplis normalement par ceux qui opèrent le logiciel.

L'objectif principal de la norme ISO/IEC 14764 consiste à fournir une directive sur «comment accomplir » le processus de maintenance et à identifier comment le processus de maintenance peut être invoqué durant la phase d'acquisition et d'opération.

La norme ISO/IEC 14764 fournit une directive pour le planning de maintenance des produits ou des services logiciels; qu'elle soit accomplie à l'intérieur ou à l'extérieur d'une organisation. Cette norme fournit une directive pour des situations à deux parties et peut être également appliquée au cas où les deux parties proviendraient de la même organisation. Elle est aussi destinée à être utilisée par une simple partie comme les tâches « s*elf-imposed* ».

Ainsi, nous remarquons que le développement de la norme ISO/IEC 14764 se réalise par discussion d'experts dans le domaine de la maintenance. Dans la section suivante, nous analysons le document de cette norme selon les critères de Vincenti.

4.2 ANALYSE DU MODÈLE DE LA NORME ISO/IEC 14764

Dans cette section, nous allons décomposer et analyser le modèle de la norme ISO/IEC 14764 sur la maintenance des logiciels. Le tableau 4.5 récapitule les différentes étapes du modèle de maintenance

de cette norme internationale selon la description de Vincenti des critères de connaissances de l'ingénierie.

Tableau 4.4 : Récapitulatif du modèle de ISO/IEC 14764 selon les critères de Vincenti

ISO/IEC 14764 Software Engineering – Software Maintenance
1. LES CONCEPTS FONDAMENTAUX DE CONCEPTION
Termes et définitions Maintenance adaptative Baseline Maintenance corrective Plan de maintenabilité Amélioration de maintenance Plan de maintenance Processus de maintenance Programme de maintenance Requête de modification Maintenance perfective Maintenance préventive Rapport de problèmes Environnement de développement du logiciel Environnement de test du logiciel Transition de logiciel
2. LES CRITÈRES ET LES SPÉCIFICATIONS
3. LES OUTILS THÉORIQUES
Stratégie de maintenance du logiciel Concept de maintenance Plan de maintenance Analyse des ressources
4. LES DONNÉES QUANTITATIVES
5. LES CONSIDÉRATIONS PRATIQUES
Considérations d'implantation Analyse des types de maintenance Arrangement pour la maintenance Outils pour la maintenance Mesures de logiciels Documentation du processus Implication précoce dans le développement Maintenabilité Transition du logiciel Documentation

ISO/IEC 14764 Software Engineering – Software Maintenance
6. LES INSTRUMENTS DE CONCEPTION
Processus de maintenance Implantation du processus Analyse du problème et de modifications Implantation de modifications Revue/acceptation de maintenance Migration du logiciel Retraite du logiciel

Dans ce tableau, nous voulons plutôt montrer une vue d'ensemble sur notre suggestion de classification des connaissances de maintenance. Cette classification s'appuie sur l'étude de la norme ISO/IEC 14764 et pose une déduction sur l'état du modèle de maintenance de cette norme. En effet, nous avons pu constaté qu'au niveau de ce modèle, il existe deux éléments qui manquent particulièrement, dans les critères de Vincenti des « critères et spécifications » et des «données quantitatives ». D'où l'insuffisance de ce modèle. Nous expliquerons davantage les détails d'une telle classification dans la section suivante.

4.2.1 Les concepts fondamentaux de conception

Nous avons évoqué dans le premier chapitre les détails sur les critères de Vincenti (p. 14) et nous avons décrit que le critère des «concepts fondamentaux de conception » consiste en des principes opérationnels associés à la configuration normale d'un système et comporte des notions et des concepts de base d'un domaine de l'ingénierie. À cet effet, nous considérons que le présent critère comporte les termes et les définitions liés à la maintenance du logiciel de la norme ISO/IEC 14764 qui sont les suivants :

Maintenance adaptative : Elle est définie comme étant la modification d'un produit logiciel, accomplie après sa livraison, pour garder utilisable le produit logiciel à un environnement changé ou à un changement d'environnement. Cette catégorie de maintenance adaptative fournit des améliorations nécessaires pour adapter le produit logiciel en question à des changements d'environnement. Les changements sont ceux qui doivent être réalisés pour garder le contrôle avec le changement d'environnement. Par exemple, le système opérant doit être amélioré et certains changements peuvent être réalisés pour adapter le nouveau système opérant.

Baseline : Il se réfère à une nouvelle livraison du logiciel. Une version approuvée formellement de l'item de configuration doit être désignée à un temps spécifique durant le cycle de vie de l'item de configuration [ISO/IEC 12207].

Maintenance corrective : Elle est définie comme étant la modification réactive d'un produit logiciel accomplie après sa livraison pour corriger les problèmes trouvés. Cette modification permet de répondre aux exigences.

Plan de maintenabilité : C'est un document permettant d'arranger les pratiques spécifiques de maintenabilité, les ressources et la séquence des activités associées à un logiciel. Ce document est préparé par le développeur.

Amélioration de maintenance : Elle est considérée comme un changement apporté à un logiciel qui n'est pas une correction de logiciel. Les améliorations d'un logiciel sont de deux types à savoir adaptative et perfective.

Plan de maintenance : C'est un document permettant d'arranger les pratiques spécifiques de maintenance, les ressources et la séquence des activités appropriées pour maintenir un produit logiciel. Ce plan est préparé par le mainteneur et doit être activé lors des transitions du produit à la phase de maintenance.

Processus de maintenance : Il contient les activités et les tâches du mainteneur. Ce processus est activé au moment où le produit logiciel subit des modifications au code ainsi que les documents associés suite à un problème ou un besoin pour l'adaptation d'amélioration. L'objectif est de modifier le produit logiciel existant tout en préservant son intégrité. Ce processus comporte la migration et la retraite du produit logiciel.

Programme de maintenance : Il représente l'infrastructure de maintenance et comporte la structure organisationnelle, les responsabilités, les procédures, les processus et les ressources utilisés pour implanter le plan de maintenance.

Requête de modification : Elle est utilisée pour identifier les changements proposés sur le produit logiciel qui est maintenu. Une requête de modification peut être classée comme une correction ou une amélioration et identifiée comme la maintenance corrective, préventive, adaptative ou perfective.

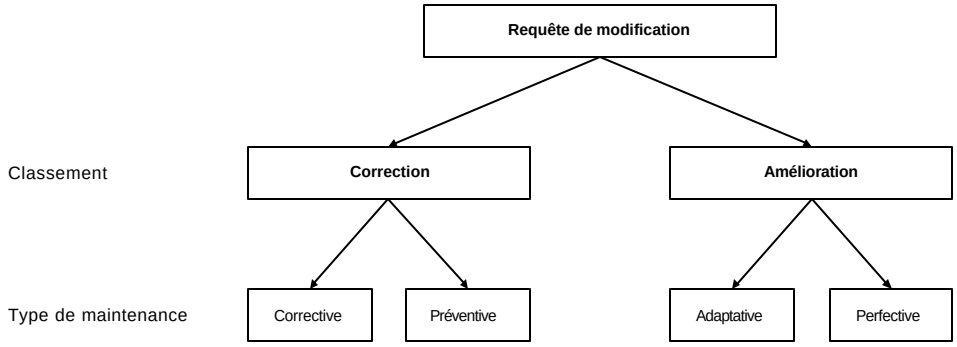

Figure 4.5 : Requête de modification [ISO/IEC 14764]

Maintenance perfective : Elle est définie comme étant la modification d'un produit logiciel après sa livraison pour améliorer la performance ou la maintenabilité. Elle fournit les améliorations pour les utilisateurs, améliorations de la documentation du programme et recodification nécessaires pour la performance, la maintenabilité, ou d'autres attributs du logiciel.

Maintenance préventive : Elle est définie comme étant la modification d'un produit logiciel après sa livraison pour détecter et corriger les défauts cachés au niveau du produit logiciel avant qu'ils deviennent effectivement des défauts.

Rapport de problèmes «*Problem Report*» : C'est un terme utilisé pour identifier et décrire les problèmes décelés au niveau d'un produit logiciel.

Environnement de développement du logiciel : Il est défini comme étant un ensemble d'outils automatisés, des dispositifs « *firmware* » et du matériel nécessaires pour accomplir l'effort du génie logiciel.

Environnement du test de logiciel :Il est constitué entre autre par des éléments portant sur les facilités matériels et logiciels ainsi que les documents nécessaires pour exécuter la qualification et le contrôle du logiciel. Ces éléments ne sont pas limités aux simulateurs, analyseurs du code et analyseurs de « *path* » mais, peuvent inclure les éléments propres à l'environnement du génie logiciel.

Transition de logiciel : Elle est définie comme étant une séquence d'actions contrôlées et coordonnées où le développement du logiciel passe de l'organisation accomplissant le développement initial du logiciel vers l'organisation accomplissant la maintenance du logiciel.

4.2.2 Les critères et les spécifications

Il ressort de l'analyse du modèle de maintenance de la norme ISO/IEC 14764 que les éléments liés au critère des « critères et spécifications » n'existent pas et que ce modèle présente des insuffisances à ce niveau.

4.2.3 Les outils théoriques

Nous incluons dans les « outils théoriques », la section de la stratégie de maintenance du logiciel telle que décrite dans la norme ISO/IEC 14764. En d'autres termes, le développement d'une stratégie de maintenance, une stratégie permettant à toutes les ressources matérielles et humaines de donner un produit logiciel de qualité. Cette stratégie comporte des notions et des concepts de base propres à la maintenance tels que les concepts de maintenance, le plan de maintenance et l'analyse des ressources. Dans une première réaction, on peut avoir tendance à placer les stratégies de maintenance dans le premier critère des « concepts fondamentaux de conception ». Mais, Vincenti met l'accent sur le fait que ce premier critère comprend « les principes opérationnels » d'un système à concevoir, alors que les stratégies de maintenance selon la norme ISO/IEC 14764 décrivent des concepts permettant de coordonner les différentes étapes de maintenance, c'est-à-dire les concepts intellectuels menant à une profonde réflexion pour bien organiser les structures de la maintenance du logiciel. D'où notre justification de mettre la section des stratégies de maintenance du logiciel dans le critère des « outils théoriques ».

Il a été énoncé dans la norme ISO/IEC 14764 que la stratégie de maintenance du logiciel doit être composée des éléments suivants :

Concept de maintenance : La détermination des concepts de maintenance devrait être la première étape dans le développement de la stratégie de maintenance du logiciel. Ces concepts devraient être développés lors de l'expression initiale des exigences d'un produit logiciel. Le concept de maintenance devrait inclure :

- la portée de maintenance du logiciel,
- l'adaptation du processus,
- la désignation de celui qui devra fournir la maintenance,
- l'estimation des coûts de maintenance.

Planning de maintenance : L'objectif du planning de maintenance consiste à planifier les activités de maintenance ainsi que les ressources utiles à la transition du produit logiciel vers la phase de maintenance. Le planning sera lancé après avoir déterminé le concept de maintenance et il sera achevé après avoir utilisé le plan de maintenance comme un guide pour les mainteneurs.

Analyse des ressources : Une fois que l'objectif de l'activité de maintenance et l'organisation accomplissant cette activité sont identifiés, il est possible de se concentrer sur la détermination du personnel, de l'environnement de maintenance et des besoins en ressources financières. Ainsi, l'acquéreur avec l'assistance du développeur détermine les besoins en ressources pour la maintenance.

4.2.4 Les données quantitatives

Bien que le rôle du critère des « données quantitatives » consiste à déterminer les détails descriptifs et prescriptifs d'un système, aucun élément n'est présent dans ce critère.

4.2.5 Les considérations pratiques

Étant donné que le critère des «considérations pratiques » représente des éléments basés sur les expériences dans la pratique (des considérations qui ne correspondent pas à des théories mais plutôt à des connaissances apprises dans la pratique), nous traiterons dans ce critère la section des considérations d'implantation décrites dans ISO/IEC 14764.

Puisqu'un produit logiciel subit des changements au cours de son cycle de vie, il est nécessaire de déterminer un processus de maintenance. Même si le développement d'un produit logiciel était effectué à l'aide des outils CASE, la maintenance est toujours nécessaire. Les outils CASE facilitent la maintenance mais n'éliminent pas le besoin pour la maintenance. Si le code d'application n'est pas développé, c'est-à-dire si le produit logiciel est composé seulement des produits « *off-the-shelf* », la maintenance est toujours requise. La maintenance des produits logiciels « *off-the-shelf* » par l'acquéreur ou le fournisseur concerne toujours la modification des interfaces, des données et des opérations du produit [ISO/IEC 14764]. Les considérations doivent être données pour compléter les

besoins et les contraintes imposés aux développeurs, mais tout en tenant compte des circonstances qui peuvent survenir et qui peuvent changer les contraintes originales.

Le processus de maintenance peut consommer une portion significative des coûts du cycle de vie. L'analyse des différents types de maintenance permet d'aider à déterminer la compréhension des coûts considérables.

Analyse des types de maintenance : La norme ISO/IEC 14764 a défini trois types de maintenance. Tout d'abord, la maintenance **corrective** qui se réfère à des changements nécessités par des erreurs réelles dans un produit logiciel. Si le produit logiciel ne répond pas aux besoins, la maintenance corrective sera accomplie. Ensuite, la maintenance **préventive** qui se réfère à des changements nécessités par la détection des erreurs possibles dans un produit logiciel. La maintenance préventive est généralement exécutée sur les produits logiciels ayant une prudence et une prévention de perdre leur vie. Enfin, le troisième type de maintenance est les changements **adaptatifs** et **perfectifs** ou en d'autre terme les **améliorations** d'un produit logiciel. Ces changements sont ceux qui ne se trouvent ni dans les spécifications de conception ni dans le logiciel libéré. Les changements adaptatifs sont des changements nécessaires pour s'adapter à un changement d'environnement. Les changements adaptatifs comportent les changements pour implanter les nouveaux besoins d'interface d'un système, les nouveaux besoins d'un système ou les nouveaux besoins matériels. Les changements perfectifs améliorent la performance ou la maintenabilité d'un produit logiciel. Un changement perfectif peut fournir de nouvelles fonctionnalités d'améliorations pour les utilisateurs ou la rétro-conception pour créer une documentation de maintenance qui n'existe pas précédemment ou pour changer une documentation existante. La maintenance du logiciel nécessite un changement au système ou à une structure existante, c'est-à-dire, les modifications d'un logiciel sont introduites dans une architecture existante et doivent répondre aux contraintes imposées par la structure de conception. Ainsi, les améliorations dans une forme de maintenance adaptative ou perfective sont souvent très coûteuses et consomment un temps considérable. Les améliorations peuvent consommer une portion significative des coûts de maintenance.

Arrangement pour la maintenance : ISO/IEC 12207 fournit les tâches détaillées pour la dérivation de l'accord entre l'acquéreur et le fournisseur. Ceci devrait être utilisé pour aider la dérivation d'un accord de maintenance si l'acquéreur ou le fournisseur font partie de la même organisation ou des organisations différentes (impartition). Si l'acquéreur demande la maintenance du logiciel auprès du développeur après la livraison, ou la fin de la période de garantie, cela devrait être spécifié dans le

contrat. La documentation mise à jour devrait être précisée dans le contrat comme un livrable. La formation également devrait être spécifiée. Ainsi, le fournisseur doit préparer les procédures pour la réalisation de maintenance et vérifier de la conformité des activités de maintenance avec les exigences spécifiées dans le contrat. À cet effet, toutes les procédures de maintenance ainsi que l'estimation du temps nécessaire à leurs réalisations doivent figurer dans le plan de maintenance.

Les procédures de modification des produits logiciels à maintenir doivent inclure :

- Les règles de base utilisées pour déclencher la modification d'un produit logiciel;
- Les descriptions des types de délivrance qui dépendent de leurs fréquences ou leurs effets sur l'opération du logiciel;
- Les manières pour lesquelles l'acquéreur devra être informé sur l'état des changements courants ou futurs;
- Les méthodes pour confirmer que les changements apportées n'induisent pas d'autres problèmes au logiciel;
- La classification des types de changement, urgence et la relation avec d'autres changements proposés.

Outils pour la maintenance : Les outils CASE peuvent être considérés comme un moyen pour accomplir les activités de maintenance du logiciel. CASE est un ensemble corrélé d'outils permettant de supporter le développement et la maintenance du logiciel [ISO/IEC 14471]. Le STE « *Software Test Environment* » permet au mainteneur de tester le produit logiciel modifié dans un environnement non-opérationnel.

Mesures de logiciels : La qualité d'un logiciel est une considération importante dans la maintenance d'un produit logiciel. Les mainteneurs doivent tenir compte des six caractéristiques de qualité de logiciel de ISO [ISO/IEC 9126]. Comme partie de mesure du logiciel, le mainteneur doit déterminer l'effort (en terme de ressources dépensées) pour les différentes catégories de maintenance. Ainsi, les données doivent être collectées, analysées et interprétées afin de faciliter l'amélioration du processus de maintenance ainsi que d'avoir une meilleure compréhension des coûts élevés de maintenance. Les données de mesures empiriques doivent être collectées afin d'aider à estimer les coûts totaux du cycle de vie du logiciel.

Documentation du processus : Le processus de maintenance du logiciel doit être bien documenté de telle façon que tout le personnel suive le même processus.

Implication précoce dans le développement : Dans la norme ISO/IEC 14764, il a été énoncé que les coûts de maintenance et la capacité du mainteneur à gérer la maintenance sont considérablement influencés par ce qui se produit lors du processus de développement du logiciel. Dans la plupart des cas pratique, le mainteneur ne peut pas être impliqué pour des raisons contractuelles ou autres. Spécifiquement, quand la maintenance est accomplie par une tierce personne, il n'y a pas une opportunité d'implication. Les fonctions accomplies par le mainteneur doivent ainsi inclure :

- Plan de logistiques pour supporter le produit logiciel,
- Assurer le support du produit logiciel,
- Supporter la planification pour la transition des produits logiciels de développement à la maintenance.

Il est à noter que le support du produit logiciel doit comporter les tâches comme le test et doit assurer la maintenabilité. La norme ISO [ISO/IEC 9126] souligne la maintenabilité et d'autres caractéristiques qui représentent des considérations importantes durant le développement. Le support peut être amélioré par la participation du mainteneur dans l'assurance-qualité, la vérification et la validation du support du processus du cycle de vie de l'ISO/IEC 12207. Ainsi, le mainteneur doit :

- participer dans les révisions,
- exécuter l'analyse du code,
- suivre les besoins,
- exécuter la vérification et la validation.

Maintenabilité : constitue un aspect important de fiabilité et une caractéristique importante du logiciel pour l'acquéreur, le fournisseur et l'utilisateur. Les besoins de maintenabilité devraient être inclus dans l'activité d'initiation du processus d'acquisition de l'ISO/IEC 12207 et devraient être évalués durant le processus de développement de l'ISO/IEC 12207. Les variations dans la conception devraient être contrôlées lors du développement pour ce qui est de l'impact sur la maintenabilité. Plusieurs mesures devraient être utilisées pour définir et évaluer la qualité du logiciel. Toutes les évaluations qualitatives et quantitatives sont importantes. La maintenabilité est une caractéristique de qualité d'un logiciel qui affecte la rapidité et la facilité de changement du logiciel après sa mise en opération [ISO/IEC 9126].

Transition du logiciel : Elle est une séquence contrôlée et coordonnée d'actions où le développement du logiciel passe de l'organisation effectuant le développement initial du logiciel vers l'organisation effectuant la maintenance du logiciel. Si la responsabilité de maintenance doit être transférée d'une organisation à une autre, il est nécessaire de développer un plan de transition. Ce plan doit souligner :

- le transfert du matériel, du logiciel, des données et de l'expérience à partir du développeur vers le mainteneur,
- les tâches nécessaires pour le mainteneur afin d'implanter la stratégie de maintenance du logiciel (exemple, personnel, formation, installation, problèmes reproduits de maintenance).

Documentation : Les mainteneurs du logiciel font souvent face à un problème de manque de documentation d'un produit logiciel. Dans le cas où la documentation n'existe pas, le mainteneur devrait créer la documentation nécessaire. Cette création fait partie de la maintenance perfective. Ceci présente des difficultés dans l'exécution de la fonction de maintenance. Dans ce cas, les mainteneurs devront suivre les étapes suivantes :

- Comprendre le domaine du problème (type d'application), éclaircir les documents (si disponibles), mener une conversation avec le développeur sur le produit logiciel (si disponible) et opérer le produit logiciel.
- Découvrir la structure et l'organisation du produit logiciel, faire un inventaire sur le produit logiciel, placer le produit logiciel sous la gestion de configuration, reconstruire le produit logiciel à partir des bibliothèques de la gestion de configuration et analyser la structure du produit logiciel.
- Déterminer la fonction du produit logiciel, réviser les spécifications (si disponibles), réviser toute la structure, analyser les arbres d'appel, comprendre le code, donner des présentations orales à d'autres mainteneurs et ajouter les commentaires au code.
- Arranger les requêtes de modification ou les rapports des problèmes par ordre de priorité.

Les mainteneurs devront documenter le produit logiciel selon la liste des étapes présentées ci-dessus. Les documents comme par exemple les spécifications, les manuels des programmeurs de maintenance, les manuels de l'utilisateur et les guides d'installation devront être mis à jour ou créés si nécessaire. Il y a plusieurs facteurs qui influencent la modification de la documentation dans un environnement de maintenance. Certains facteurs comportent : l'accès au code source, la disponibilité des outils pour analyser le code, la compétence d'opérer le produit logiciel en déterminant les capacités et la disponibilité du STE (*Software Test Environment*).

4.2.6 Les instruments de conception

Ce critère concerne réellement la manière de réaliser les tâches et «comment » procéder à la maintenance. Cela sera traité dans la section du processus de maintenance.

Le processus de maintenance contient les activités et les tâches nécessaires pour modifier un produit logiciel existant tout en gardant son intégrité. Ces activités représentent les responsabilités du mainteneur. La norme ISO/IEC 14764 fournit les étapes en se référant à des exemples démonstratifs

qui permettent d'implanter les activités de maintenance. Le mainteneur devrait s'assurer de l'existence du processus de maintenance lors du développement du produit logiciel. Le processus de maintenance devrait être activé quand il y a un besoin de maintenir le produit logiciel. Une fois que le processus sera activé, les mainteneurs devront modifier le code ainsi que les documents associés pour répondre aux exigences d'une requête de modification ou d'un rapport de problèmes. L'objectif derrière une activité de maintenance est de modifier le produit existant tout en gardant son intégrité.

Le processus de maintenance supporte le produit logiciel depuis sa mise en opération jusqu'à sa retraite. Ce processus se termine quand le produit logiciel est finalement abandonné. Il est constitué des activités suivantes :

1 Implantation du processus,
2 Analyse du problème et des modifications,
3 Implantation de modification,
4 Revue/acceptation de maintenance,
5 Migration du logiciel,
6 Abandon du logiciel.

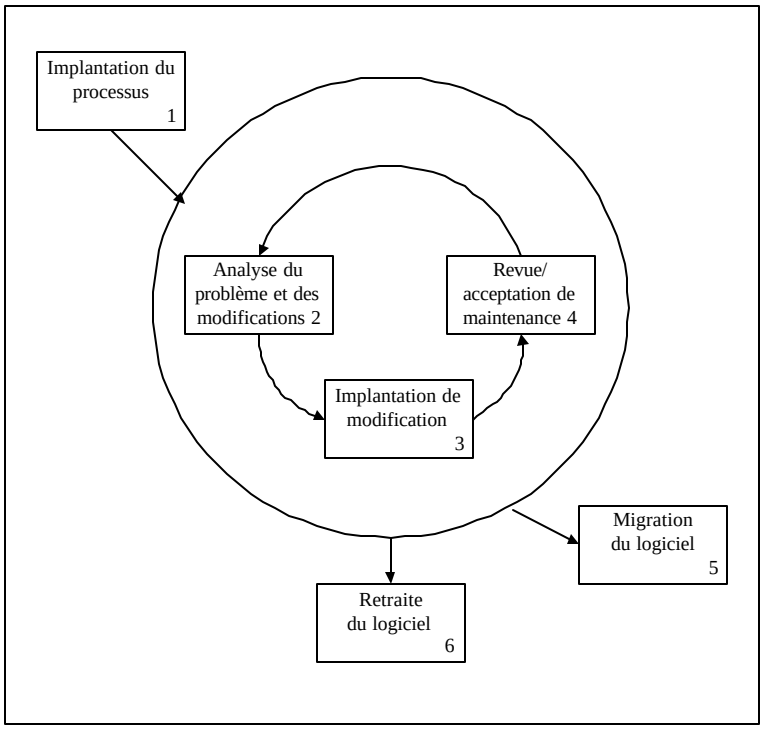

Figure 4.6 : Processus de maintenance (Source ISO/IEC 14764)

Implantation du processus : Au cours de cette étape, le mainteneur devrait établir les plans et les procédures qui vont être exécutés durant le processus de maintenance. Le plan de maintenance devrait être développé en parallèle avec le plan de développement. Le mainteneur devrait également établir les interfaces organisationnelles lors de cette activité.

Analyse du problème et des modifications : Lors de cette activité, le mainteneur devrait :

- analyser les requêtes de modification,
- copier ou vérifier le problème,
- développer des options pour l'implantation des modifications,
- documenter les requêtes de modification ou le rapport des problèmes, les résultats et les options d'implantation,
- obtenir le consentement pour l'option de modification sélectionnée.

Implantation de modification : Au cours de cette activité, le mainteneur doit développer et tester les modifications apportées à un produit logiciel.

Revue/acceptation de maintenance : Cette activité permet de s'assurer que les modifications réalisées dans un système logiciel sont correctes et qu'elles sont conformes aux normes approuvées en utilisant la méthodologie de correction.

Migration du logiciel : Lors de sa vie, un système logiciel peut subir des modifications pour s'exécuter dans différents environnements. Afin de migrer un système à un nouvel environnement, le mainteneur devrait déterminer les actions utiles pour accomplir cette migration. Par la suite, il devrait développer et documenter les étapes nécessaires à la réalisation de la migration.

Retraite du logiciel : Une fois que le produit logiciel a atteint la fin utile de sa vie, il doit être retiré. Une analyse devrait être réalisée pour aider à prendre la décision d'abandonner un produit logiciel. L'analyse est souvent effectuée sur des bases économiques et peut être incluse dans le plan de retraite du logiciel. L'analyse devrait déterminer si son coût est élevé pour :

- retenir la technologie désuète,
- changer vers une nouvelle technologie en développant un nouveau produit logiciel,
- développer un nouveau produit logiciel pour atteindre la modularité,
- développer un nouveau produit logiciel pour faciliter la maintenance,
- développer un nouveau produit logiciel pour atteindre la standardisation,
- développer un nouveau produit logiciel pour être indépendant du vendeur.

Le produit logiciel peut être remplacé par un nouveau produit logiciel mais dans la plupart des cas, il ne sera pas remplacé. Afin d'abandonner un produit logiciel, le mainteneur devrait déterminer les actions nécessaires pour accomplir la retraite du logiciel et développer et documenter ainsi les étapes requises pour effectuer cette retraite. Des considérations devront être fournies pour accéder aux données sauvegardées par un produit logiciel retiré.

4.3 CONCLUSION

La faiblesse principale réside dans le fait que le modèle de maintenance de la norme ISO/IEC 14764 a été développé par discussion d'experts et n'a pas été vérifié par d'autres que les participants de cette discussion. Pour cela, on retrouve dans la littérature des types de maintenance ayant la même fonction que ce qui a été énoncé dans la norme ISO/IEC 14764 mais portant des noms différents, exemple, dans les travaux de Pigoski on retrouve le vocabulaire de maintenance corrective correspondant au groupement correction de ISO/IEC 14764. De plus, le modèle ISO/IEC 14764 prescrit ce qu'un mainteneur doit faire plutôt que la diversité des choses que ce mainteneur peut ou pourrait faire. En effet, ceci est une limite de toutes les normes qui, par définition, sont prescriptives. Mais, **i** y a d'autres concepts de maintenance qu'on retrouve dans la littérature. Par exemple, dans le livre de Pigoski, on retrouve un aspect pratique, c'est-à-dire ce que le mainteneur doit faire.

Malgré ses limites, la norme ISO nous renseigne cependant assez bien sur les propriétés pratiques des concepts de maintenance par rapport au mainteneur, c'est-à-dire sur la manière dont on peut accomplir la maintenance des logiciels.

Le modèle de maintenance de la norme ISO/IEC 14764 n'est pas suffisant ni assez mature. Les faiblesses de ce modèle telles qu'évaluées à l'aide de la classification de Vincenti, concernent deux critères : l'un est relié aux «critères et spécifications », celui qui nous permet de traduire les buts qualitatifs d'un système logiciel (le général) en des buts quantitatifs sous forme de termes techniques concrets (le spécifique), l'autre est relié aux «données quantitatives », celui qui nous permet de déterminer les détails d'un système à concevoir.

L'analyse dans ce chapitre nous permet de procéder à l'analyse comparative des divers modèles de maintenance de SWEBOK, du texte de Pigoski et de la norme ISO/IEC 14764.

Le chapitre suivant présente l'étude comparative sur les connaissances de maintenance afin de proposer des suggestions d'amélioration à ce modèle.

CHAPITRE V

ÉTUDE COMPARATIVE DES MODÈLES DE ISO/IEC 14764, DE SWEBOK ET DES TRAVAUX DE PIGOSKI

Nous aborderons, dans ce chapitre, une étude comparative des différents modèles du projet SWEBOK, des travaux de Pigoski et de la norme ISO/IEC 14764 sur la maintenance tels qu'évalués à l'aide des critères de Vincenti. Nous examinerons, dans le contexte de l'ingénierie du logiciel, les points particuliers que nous rencontrons dans ces modèles pour apporter des suggestions d'amélioration aux divers modèles portant sur la maintenance des logiciels.

5.1 SYNTHÈSE ET COMPARAISON ENTRE SWEBOK, TRAVAUX DE PIGOSKI ET ISO

Tout d'abord, afin que notre analyse comparative soit menée à bien, tous les éléments identifiés des critères de Vincenti devraient être présents. En d'autres termes, dans un contexte de l'ingénierie, un domaine de connaissance est considéré mature lorsque tous ses éléments ou les sous-domaines de connaissances associés sont présents selon les critères de la classification de Vincenti des connaissances de l'ingénierie. Ainsi, un modèle de maintenance serait considéré mature à partir du moment où il répondrait à tous les critères de Vincenti. Lorsqu'un élément manque, on peut en déduire que le domaine de connaissance en question présente encore des insuffisances importantes et, par conséquent, doit être considéré comme étant un domaine immature. Le tableau 5.6 résume les modèles de maintenance qui ont été analysés et nous en dégageons qu'il y a plusieurs insuffisances et faiblesses dans ces divers modèles.

Tableau 5.5 : Récapitulatif des divers modèles de maintenance selon les critères de Vincenti

SWEBOK (Version 0.7)	Travaux de Pigoski	ISO/IEC 14764
1- LES CONCEPTS FONDAMENTAUX DE CONCEPTION		
Maintenance du logiciel : • Besoin pour la maintenance • Catégories de maintenance Activités de maintenance : • Activités uniques • Activités de support • Activités de planification de maintenance Aspect d'organisation de maintenance : • Mainteneur • Impartition • Structure organisationnelle	Maintenance du logiciel Activité pré-livraison de maintenance du logiciel Planning : • Concept de maintenance • Plan de maintenance • Ressources Organisations de la maintenance du logiciel	Termes et définitions : • Maintenance adaptative • Baseline • Maintenance corrective • Plan de maintenabilité • Amélioration de maintenance • Plan de maintenance • Processus de maintenance • Programme de maintenance • Requête de modification • Maintenance perfective • Maintenance préventive • Rapport de problèmes • Environnement de développement du logiciel • Environnement de test du logiciel • Transition de logiciel
2- LES CRITÈRES ET LES SPÉCIFICATIONS		
Mesures de maintenance des logiciels : • Établissement d'un programme de mesure • Mesures spécifiques	Mesures de maintenance des logiciels	
3- LES OUTILS THÉORIQUES		
		Stratégie de maintenance du logiciel : • Concept de maintenance • Plan de maintenance • Analyse des ressources
4- LES DONNÉES QUANTITATIVES		
Coût de maintenance et estimation des coûts de maintenance : • Coût • Estimation des coûts	Expériences de mesures de maintenance du logiciel	

SWEBOK (Version 0.7)	Travaux de Pigoski	ISO/IEC 14764
5- LES CONSIDÉRATIONS PRATIQUES		
Problèmes de maintenance du logiciel : *Problèmes techniques :* • Compréhension limitée • Test • Analyse de l'impact • Maintenabilité *Problèmes de gestion :* • Alignement avec les problèmes organisationnels • Personnel • Problème de processus Techniques pour la maintenance • Compréhension du programme • Réingénierie • Rétro-conception • Analyse de l'impact Ressources	Transition Outils et environnement Maintenabilité Gestion de maintenance du logiciel Éducation et formation Impact de la technologie orientée objet sur la maintenance	Considérations d'implantation : • Analyse des types de maintenance • Arrangement pour la maintenance • Outils pour la maintenance • Mesures de logiciels • Documentation du processus • Implication précoce dans le développement • Maintenabilité • Transition du logiciel • Documentation
6- LES INSTRUMENTS DE CONCEPTION		
Processus de maintenance : • Modèles du processus de maintenance	Processus de maintenance de l'ISO/IEC 12207 : • Implantation du processus • Analyse du problème et des modifications • Implantation des modifications • Revue/acceptation de la maintenance • Migration • Retraite du logiciel	Processus de maintenance de l'ISO/IEC 12207 : • Implantation du processus • Analyse du problème et des modifications • Implantation des modifications • Revue/acceptation de maintenance • Migration du logiciel • Retraite du logiciel

Ainsi pour le domaine de connaissance de la maintenance du logiciel, nous remarquons que tous les modèles (SWEBOK, les travaux de Pigoski et l'ISO) sont incomplets par rapport aux critères de connaissance en ingénierie de Vincenti. Cependant, les insuffisances de l'un ne sont pas nécessairement les insuffisances de l'autre. Par exemple, dans SWEBOK et dans le texte de Pigoski, il y a un vide dans le critère des « outils théoriques » alors que dans la norme ISO/IEC 14764, il y a un manque d'éléments aussi bien dans le critère des «critères et des spécifications » que celui des « données quantitatives ». De même, on ne peut pas dire qu'il y a une complémentarité, car chacun de ces sujets comporte une structure particulière qui dépend étroitement de la perspective choisie par l'auteur. Les différences que nous avons remarqué entre les divers sujets dans les modèles sur la

maintenance de la norme ISO/IEC 14764, du projet SWEBOK (version 0.7) et des travaux de Pigoski sont décrits dans la section suivante.

5.1.1 Premier critère de comparaison : Les concepts fondamentaux de conception

1. Différences au niveau des concepts de base :

Avec l'analyse des différents textes portant sur la maintenance du point de vue de l'ingénierie, nous avons remarqué que les concepts ou les représentations pour définir la maintenance ne sont pas semblables. En effet, dans la norme la ISO/IEC 14764, la maintenance est définit comme étant la modification du code et des documents associés dans un objectif d'amélioration ou de correction et que cette fonction de maintenance devrait être considérée avant la livraison du produit logiciel, c'est-à-dire lors du planning. Dans le projet SWEBOK, la maintenance est plutôt définit comme étant la totalité des activités requises pour fournir un support coût-effectif du système logiciel. Ainsi, les terminologies de la norme ISO impliquent une généralisation à partir des concepts qui ont été perçus ou constatés par les experts du domaine. Ceux du projet SWEBOK, représentent d'une part ce qui est général, c'est-à-dire commun ou applicable à plusieurs situations, et ce qui est abstrait, c'est-à-dire séparé de certains éléments auxquels sont liés dans la réalité concrète (connaissances fournies par les académiciens et les chercheurs) d'autre part.

2. Différences au niveau des catégories de maintenance :

Comparativement aux concepts de base, les définitions des catégories de maintenance sont générales et abstraites. Dans les différents textes, on retrouve des noms différents portant la même signification. Ainsi, comme nous l'avons vu dans le chapitre 2, Swanson de UCLA fut le premier chercheur qui a examiné la fonction de maintenance du logiciel et qui a regroupé cette fonction en trois catégories selon les causes constatées : maintenance corrective, maintenance adaptative et maintenance perfective. Par la suite, ces catégories ont été modifiées par les normes ISO/IEC 14764 et IEEE 1219 avec l'ajout d'une quatrième catégorie : maintenance préventive. De plus, la norme ISO/IEC 14764 classifie les catégories de maintenance adaptative et perfective comme des améliorations et les catégories de maintenance correctives et préventives comme des corrections. La maintenance préventive est souvent accomplie sur le produit logiciel lorsque le facteur « sécurité » est critique.

Dans le projet SWEBOK, on retrouve les quatre types de maintenance telles qu'elles sont définies par la norme ISO/IEC 14764. On retrouve également que la maintenance préventive peut facilement être incluse dans la catégorie de maintenance corrective mais la communauté internationale la classifie comme étant un type séparé de maintenance.

Pigoski, quant à lui, propose de garder deux groupes de maintenance : la maintenance corrective et les améliorations qui sont la fusion de la maintenance perfective et la maintenance adaptative.

Les définitions des différentes catégories de maintenance nous apportent une information réelle sur les catégories de maintenance, mais, il y a une ambiguïté au niveau du vocabulaire et les terminologies ne sont pas bien uniformisées. En effet, la maintenance des logiciels, telle qu'elle est identifiée dans la norme ISO, comporte deux groupes. L'un est le groupe « amélioration » correspondant aux catégories de maintenance adaptative et perfective. L'autre est le groupe «correction » correspondant aux catégories de maintenance corrective et préventive. D'autre part, dans la littérature de Pigoski, on retrouve que la catégorie de maintenance préventive n'est pas conservée ou encore qu'elle est introduite dans la catégorie de maintenance corrective. À cet effet, on peut dire que les diverses catégories de maintenance ne sont pas conçues de la même façon. Elles sont pratiquement traduites par des représentations générales et abstraites. Ces représentations ne peuvent être matérialisées que par des discussions qui nomment les propriétés communes à une catégorie donnée. En d'autres termes, les discussions se réalisent entre les communautés techniques des normes ou entre les académiciens ou entre les chercheurs du domaine pour matérialiser les pensées et les concepts.

Nous avons remarqué la succession de la définition des catégories de maintenance allant du document de Swanson (1989) vers le document de la norme ISO/IEC 14764 et celui du projet SWEBOK.

Cette succession est due principalement à des problèmes reliés aux définitions des besoins et d'anticipation des problèmes qui peuvent survenir. La formulation des définitions nécessite non seulement des connaissances conceptuelles mais aussi une synthèse des perceptions et des choses constatées dans la pratique. L'anticipation des problèmes nécessite de prendre des mesures, exprimées en terme de maturité, pour matérialiser et caractériser le domaine en question. Ainsi, on peut dire que les normes sont nécessaires mais pas suffisantes pour construire le domaine du génie logiciel.

La notion de terminologie devrait être considérée comme un sujet à part dans la discipline du génie logiciel pour résoudre les problèmes d'ambiguïté et arriver à avoir un vocabulaire uniforme entre les académiciens, les praticiens et les chercheurs.

3. Différences au niveau des activités liées à la maintenance :

D'après l'analyse des divers modèles en ce qui concerne les activités de maintenance, on peut dégager que la distinction entre le développement et la maintenance des logiciels n'est pas bien reflétée. Dans le projet SWEBOK, il y a une ambiguïté dans les expressions à ce niveau. D'une part, on retrouve que la maintenance est similaire au développement, et d'autre part, elle est différente du développement. La signification et le sens des termes « similaire » et « différente » ne sont pas bien identifiés. Dans le livre de Pigoski, l'accent se fait plutôt sur les changements que le mainteneur doit entreprendre à travers les étapes pré-livraison et post-livraison du logiciel. Étant donné que ces changements diffèrent d'une organisation à une autre selon le processus d'exécution, les activités de maintenance n'ont pas un contenu et un statut normalisés. Par contre dans la norme ISO/IEC 14764, on retrouve que les activités de base d'un processus de maintenance sont bien décrites.

Nous avons identifié ainsi trois différences dans ce premier critère de comparaison des « concepts fondamentaux de conception ». La première concerne les concepts de base liés à la maintenance. La définition de ces concepts, nous le rappelons, n'est pas similaire dans tous les modèles et ne reflète pas exactement la même signification. La seconde qui a rapport aux catégories de maintenance soulève le problème d'ambiguïté dans le vocabulaire. La troisième différence concerne les activités de maintenance et présente également le problème d'ambiguïté dans le vocabulaire et la définition des termes nécessaires pour montrer ce qu'ils signifient par rapport à ce qu'ils ne signifient pas.

5.1.2 Deuxième critère de comparaison : Les critères et les spécifications

1. Différences au niveau des mesures de maintenance des logiciels :

Ce deuxième critère de comparaison soulève entre autres des problèmes similaires au premier. En effet, nous avons remarqué une ambiguïté dans l'utilisation du terme « mesure » aussi bien dans le projet SWEBOK que dans les travaux de Pigoski. En ce sens, on retrouve les termes « mesure » et « métrique » qui désignent la même chose et qui permettent de traduire les buts qualitatifs d'un système en des buts quantitatifs concrets. Par ailleurs, dans la norme ISO/IEC 14764 l'aspect mesure a

été identifié dans la catégorie « considérations pratiques » du point de vue contrôle et gestion pour aboutir à la mesure de la qualité d'un produit logiciel, et non pas du point de vue spécification quantitative. C'est pourquoi on y retrouve des insuffisances dans le critère des « critères et spécifications ». Dans cette norme, la conception et la définition des mesures sont entreprises d'une perspective gestion.

Dans le vocabulaire des scientifiques, les termes acceptés sont « *measure* » et « *measurement* ». Le terme « métrique » constitue principalement un « *buzzword* » non accepté en sciences, même s'il est couramment utilisé dans les articles publiés en génie logiciel et il n'y a pas de consensus sur la définition de « métrique ». C'est pour cela, il a été préconisé de garder le vocabulaire accepté en métrologie classique, soit le terme « mesure » au lieu de « métrique » [Abran et Jacquet, 1999].

5.1.3 Troisième critère de comparaison : Les outils théoriques

Bien que les « outils théoriques » représentent pour les ingénieurs un moyen pour accomplir leurs fonctions et pour décrire les concepts intellectuels, les méthodes mathématiques et les théories des calculs conceptuels, aucun élément n'a été identifié dans les modèles de SWEBOK et des travaux de Pigoski au niveau du critère des « outils théoriques ». À l'inverse, la norme ISO/IEC 14764 présente des forces à ce niveau. Cette norme décrit les concepts intellectuels permettant de coordonner les différentes étapes de maintenance ainsi que les théories de base pour procéder à une démarche de maintenance ayant une structure explicative. Vincenti énonce que « Bien que l'activité de l'ingénierie consiste à analyser, concevoir et produire des objets, elle nécessite aussi des réflexions humaines ». En ce sens, le présent critère devrait comporter des éléments pour organiser les structures de maintenance.

Ainsi, nous remarquons qu'à travers les outils théoriques, on peut réussir à expliquer des faits concrets de la pratique. En d'autres termes, on peut recourir à des causes pour rendre compte d'une bonne stratégie de maintenance.

5.1.4 Quatrième critère de comparaison : Les données quantitatives

Comparativement au critère précédent, les connaissances de cette catégorie sont plus ou moins tangibles. Dans son texte, Vincenti renforce l'importance des données dans l'ingénierie et montre que les données quantitatives peuvent être dérivées théoriquement ou empiriquement et peuvent être soient descriptives ou prescriptives. Notre étude révèle que les propriétés de ces données ne sont pas toutes présentes dans les modèles de maintenance. En effet, dans les deux modèles du projet SWEBOK et du

texte de Pigoski, on retrouve des concepts corroborés par l'expérience et les observations empiriques dans les «données quantitatives ». On vérifie les concepts en passant par la vérification de la compatibilité avec les connaissances déjà admises et la vérification expérimentale. Mais, il arrive qu'on déduise logiquement des concepts des données qui n'ont pas encore été connues par la voie expérimentale et qu'on observe par la suite, après une recherche expérimentale suggérée par les concepts (par exemple, les connaissances prescriptives). Ce qui est le cas pour la norme ISO. Celle-ci n'est pas basée directement sur des observations empiriques; elle résulte plutôt des travaux d'individus qui se ressemblent et forment des groupes de travail pour son établissement.

5.1.5 Cinquième critère de comparaison : Les considérations pratiques

S'appuyant sur les modèles présentés dans le tableau 5.6, nous déterminons les différences au niveau du critère des « considérations pratiques ». Ce critère repose sur les expériences dans la pratique qui sont susceptibles de changer. Vincenti vise par l'utilisation de ce critère d'extraire les connaissances basées plus sur l'expérience que sur les habilités. Elles comportent ainsi les éléments fondamentaux de l'expérience du mainteneur et les règles du pouce (« rules of *thumb* ») d'une expérience précédente.

Dans le vocabulaire de la norme ISO/IEC 14764, les « considérations pratiques » désignent un énoncé décrivant les considérations d'implantation nécessaires pour compléter les contraintes imposées précédemment aux développeurs et tenir compte des circonstances qui peuvent survenir et qui peuvent changer les contraintes originales.

Dans le sens du projet SWEBOK, les considérations pratiques désignent l'ensemble des problèmes d'ordre technique et de gestion qui peuvent survenir lors de la maintenance. Ces problèmes sont introduits pour s'assurer de leurs généralisations totales. Les « considérations pratiques » désignent également l'ensemble des techniques généralement acceptées et utilisées dans les opérations de maintenance ainsi que les ressources appropriées.

Dans son livre, Pigoski nous indique les meilleures approches pratiques liées à la maintenance des logiciels. Nous retrouvons ainsi dans le critère des « considérations pratiques » l'aspect technique comportant l'impact de la technologie orientée objet et l'aspect gestion sur la maintenance.

Les éléments fournis dans les « considérations pratiques » permettent ainsi de diriger et accomplir plus efficacement les activités cognitives, connaître et bien maîtriser les méthodes et les concepts fondamentaux liés à la maintenance.

5.1.6 Sixième critère de comparaison : Les instruments de conception

Il découle du tableau 5.6 sur le critère des « instruments de conception » que les divers modèles possèdent un caractère commun qui est le processus de maintenance du logiciel de la norme ISO/IEC 12207. Vincenti a inclus dans ce critère les procédures structurées (comme la décomposition d'un problème en sous-problèmes), les façons de penser et d'accomplir les tâches (exemple, l'utilisation de l'analogie et les approches de « ce que devrait se produire? ») et les habilités de jugement (exemple, la capacité de comparer les conflits en besoins de conception), [Réf. Vincenti, p. 200]. Tous les modèles s'entendent sur le fait que le processus de ISO/IEC 12207 constitue un instrument approprié pour les mainteneurs en tenant compte des habilités de jugement et des compétences de ces derniers. Les spécifications de ce critère présentent une vision plus large dans le projet SWEBOK. Ce dernier, ajoute l'introduction du modèle CMM (*Capability Maturity Model*) du SEI (*Software Engineering Institute*) comme un instrument de mesure des niveaux de maturité d'un processus logiciel et un moyen permettant d'éviter les problèmes d'incohérence.

En conclusion, dans le domaine de maintenance du point de vue de l'ingénierie, le vocabulaire doit être choisi avec soin. Dans le sens que chaque terme doit avoir une signification pour tous les membres du domaine. Le vocabulaire doit également exprimer le concept de maintenance, sans entrer en conflit avec le langage des praticiens ou des académiciens.

Compte tenu de ce qui a été dit plus haut sur la façon de décrire les divers modèles de maintenance, nous remarquons que les connaissances de maintenance dépendent étroitement des expériences pratiques et peuvent commencer par la généralisation des choses qui ont été perçues ou constatées. En d'autres termes, on peut aboutir à former des concepts à partir des perceptions. Mais, à l'inverse, les concepts peuvent influencer les perceptions, c'est-à-dire la façon de percevoir : les connaissances déjà acquises nous amènent à percevoir autrement les choses et à les interpréter. C'est une des raisons pour lesquelles on retrouve des points de vue différents et des terminologies différentes entre les académiciens et les gens de l'industrie.

En se basant sur les différences présentées ci-dessus et sur le fait que Pigoski est l'auteur du projet SWEBOK sur la maintenance, nous allons nous limiter dans ce qui suit, aux modèles de maintenance de ISO/IEC 14764 et de SWEBOK (version 0.7). Nous allons identifier les forces et les faiblesses de ces modèles pour aboutir à des suggestions d'amélioration.

5.2 FORCES ET FAIBLESSES DE LA NORME ISO/IEC 14764

Dans la norme ISO/IEC 14764, la maintenance du logiciel est entreprise d'une perspective gestion pour fournir un guide de planification des produits et services de maintenance. La portée de cette norme ISO vise à répondre aux questions « Comment accomplir le processus de maintenance? » ou « Que doit faire le mainteneur? ». Cependant, il manque de fournir « la diversité des choses que le mainteneur peut ou pourrait faire ». Par exemple, elle définit les termes et les définitions de maintenance du logiciel et prescrit un guide pour le processus de maintenance mais ne fournit pas les besoins d'une perspective computationnelle et appliquée du traitement cognitif de l'information provenant des praticiens dans la réalisation de la maintenance.

Tableau 5.6 : Sommaire des forces et faiblesses de ISO/IEC 14764

Modèle de Maintenance du logiciel	**Document ISO/IEC 14764**	
	Forces	**Faiblesses**
1. Concepts fondamentaux de conception		
Concepts de base	Prescrit les concepts de maintenance du logiciel avant la livraison du produit logiciel.	Prescrit les concepts seulement du point de vue pratique et qui sont constatés généralement par les experts du domaine.
Catégories de maintenance	Définit deux groupes de catégories de maintenance : les améliorations (adaptative et perfective) et les corrections (corrective et préventive).	
Activités de maintenance	Définit les activités et les tâches du mainteneur dans le processus de maintenance.	Les activités spécifiques à la maintenance ne sont pas explicitement soulignées.
2. Critères et spécifications		
		Pas de spécifications des besoins spécifiés. La traduction des buts qualitatifs d'un système en des buts quantitatifs concrets n'est pas bien identifiée

Modèle de Maintenance du logiciel	Document ISO/IEC 14764	
	Forces	**Faiblesses**
3. Outils théoriques		
	Décrit les concepts intellectuels pour coordonner les différentes étapes de la maintenance.	
4. Données quantitatives		
		N'adresse pas des observations empiriques.
5. Considérations pratiques		
	Fournit les considérations d'implantation du processus de maintenance.	
6. Instruments de conception		
	Prescrit un guide sur la gestion du processus de maintenance.	

5.3 FORCES ET FAIBLESSES DU MODÈLE DE MAINTENANCE DE SWEBOK

Comparativement à la norme ISO, le modèle du projet SWEBOK sur la maintenance du logiciel vise à identifier les connaissances sur la maintenance en tenant compte des points de vue des spécialistes dans les connaissances, des praticiens et des chercheurs dans le domaine. L'étendue de ce projet se limite à fournir des documents de référence pour enrichir les connaissances en génie logiciel et traiter ainsi la maintenance du logiciel dans un contexte plus large que celui des normes.

Tableau 5.7 : Sommaire des forces et faiblesses de SWEBOK

Modèle de Maintenance du logiciel	Document SWEBOK (Version 0.7)	
	Forces	**Faiblesses**
1. Concepts fondamentaux de conception		
Concepts de base	Prescrit les concepts de maintenance du logiciel en tenant compte des points de vue des spécialistes et des chercheurs praticiens dans le domaine.	
Catégories de maintenance	Spécifie les quatre catégories de maintenance telles que décrites dans les normes et les travaux de Pigoski.	
Activités de maintenance	Prescrit la liste des activités de maintenance du logiciel.	La définition des activités de maintenance par rapport au développement du logiciel est ambiguë.
2. Critères et spécifications		
	Fournit les stratégies de mesure de logiciel et le manque de mesures spécifique à la maintenance.	
3. Outils théoriques		
		Ne décrit pas explicitement les concepts intellectuels pour coordonner les étapes de maintenance.
4. Données quantitatives		
	Spécifie l'aspect coût de maintenance et l'estimation des coûts de maintenance.	
5. Considérations pratiques		
	Met en relief les problèmes qui apparaissent lors de la réalisation de maintenance.	
6. Instruments de conception		
	Décrit le processus de maintenance de la norme ISO/IEC 12207. Décrit le modèle CMM comme moyen permettant d'éviter l'incohérence.	

5.4 CONTRAINTES ET SUGGESTIONS D'AMÉLIORATION

Tout d'abord, il est important de rappeler que la classification des connaissances de l'ingénierie de Vincenti, sur laquelle nous nous sommes basés dans notre étude, était identifiée dans le domaine physique du génie aéronautique et particulièrement dans la phase de conception d'un système. L'application de cette classification à la maintenance du logiciel du point de vue de l'ingénierie est possible malgré le caractère abstrait de ce domaine et la diversité de descriptions de la maintenance.

Au niveau des communautés de normalisation, nous suggérons dans le tableau qui suit de concilier entre la théorie et la pratique :

Tableau 5.8 : Proposition de modèle amélioré de maintenance de ISO/IEC 14764

1. Concepts fondamentaux de conception	La création des concepts doit se baser sur les points de vue des experts et sur des observations systématiques dans un large éventail de contextes industriels et académiques : • Spécifier le planning de maintenance pour chaque aspect de l'organisation de celle-ci (mainteneur, impartition, etc.) ainsi que les activités de pré-livraison et de post-livraison [SWEBOK, 2000]. En effet, ISO/IEC met l'accent seulement sur les activités pré-livraison. Un glossaire contenant le vocabulaire international des termes semble nécessaire pour prescrire l'équivalence des termes ayant le même nom mais qui ne définissent pas la même chose : • Les catégories de maintenance sont indispensables pour comprendre sa signification principale [IEEE, 1219] : la maintenance préventive est définie par les normes dans l'objectif de prévoir les problèmes avant qu'ils n'apparaissent [SWEBOK, 2000]. D'autre part, elle consiste à détecter et corriger les erreurs en latence avant qu'elles deviennent des erreurs effectives.
2. Critères et spécifications	La description explicite des critères techniques et la spécification des besoins doivent être spécifiés pour fixer les attributs de la qualité de la maintenance : • À l'inverse du modèle ISO/IEC 14764, SWEBOK identifie des mesures spécifiques à la maintenance [Pigoski, 1997].

3. Outils théoriques	Dans cette norme ISO, ce critère est explicitement décrit. En effet, on retrouve la stratégie de maintenance du logiciel comportant : • Concept de maintenance, • Plan de maintenance, • Analyse des ressources.
4. Données quantitatives	En génie logiciel, ces données ne sont pas faciles à identifier; les concepts sont abstraits mais, il convient de spécifier les mesures ou l'estimation de maintenance dans ce critère pour avoir plus de détails sur le système à maintenir [SWEBOK, 2000].
5. Considérations pratiques	Dans ce critère, on retrouve les considérations d'implantation mais il manque la présence des principaux problèmes liés à la maintenance du logiciel, à savoir les problèmes techniques et de gestion [Pfleeger, 1998 ; SWEBOK, 2000].
6. Instruments de conception	Le guide de processus de maintenance est bien spécifié dans ce critère. Reste à préciser les activités propres à la maintenance pour différencier clairement le développement par rapport à la maintenance du logiciel.

Si nous nous référons aux dates de publication des différents modèles de maintenance de ISO/IEC 14764, de Pigoski et de la version 0.7 du projet SWEBOK (indiqué dans la figure 5.7), nous pouvons dire qu'il est évident de trouver des améliorations pour le modèle ISO. D'autre part, un objectif important de notre projet était d'apporter des suggestions d'amélioration à SWEBOK, ce qui était un défi plus grand puisque ce dernier document est très récent et évoluait en parallèle à notre propre projet d'analyse et de comparaison. Nous les présentons de façon sommaire dans le tableau qui suit.

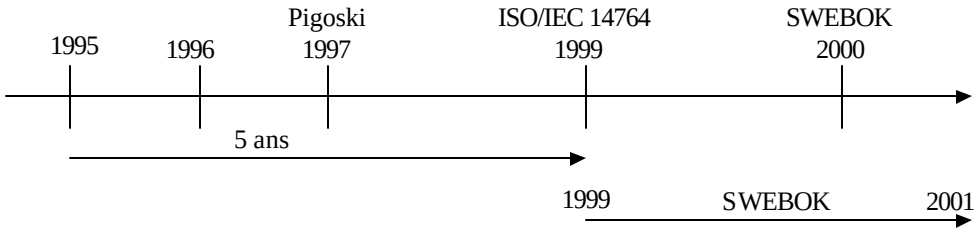

Figure 5.7 : Succession des divers modèles de maintenance

Dans le cadre du projet SWEBOK, nous proposons d'introduire dans le critère des « outils théoriques » les concepts nécessaires pour bien organiser les structures de maintenance et coordonner ainsi les différentes étapes de maintenance, tels que les stratégies de maintenance. Les faiblesses que nous avons identifiées dans la version 0.7 de ce projet ont été rectifiées dans la version 0.9.

Tableau 5.9 : Proposition de modèle amélioré de maintenance de SWEBOK

1. Concepts fondamentaux de conception	Les activités de maintenance ne doivent pas figurées dans ce critère. En effet, la structure de SWEBOK a été améliorée dans la version_0.9.; les activités de base sont décrites dans le processus de maintenance.
2. Critères et spécifications	Dans ce critère, on retrouve les mesures de maintenance des logiciels. Mais, la sous-section portant sur l'établissement d'un programme de mesure devrait être dans la catégorie des « considérations pratiques ».
3. Outils théoriques	Il semble nécessaire d'introduire la stratégie de maintenance dans ce critère comme décrite dans la norme ISO pour organiser les structures de maintenance [ISO/IEC 14764].
4. Données quantitatives	Ce critère est bien défini dans SWEBOK (version 0.7).
5. Considérations pratiques	L'aspect pratique est bien décrit. L'introduction des considérations d'implantation comme décrite dans ISO semble nécessaire.
6. Instruments de conception	Ce critère est bien défini dans SWEBOK (version 0.7).

CONCLUSION

Ce mémoire se situe dans le cadre des travaux de recherche longitudinale sur le problème de maturité des modèles de maintenance du logiciel dans une perspective de l'ingénierie. La maintenance des logiciels reste toujours la plus dispendieuse dans le cycle de vie du logiciel. Les documents de synthèse sur la maintenance sont des modèles qui ne sont pas conçus de la même façon entre les académiciens et les praticiens. Partant de ces observations, notre contribution pour ce projet a consisté à élaborer un rapport comportant des suggestions d'amélioration de ces modèles.

Notre recherche est particulièrement basée sur les modèles de maintenance de la norme ISO/IEC 14764, du projet SWEBOK et de les travaux de Pigoski pour concilier entre la théorie et la pratique.

Au plan méthodologique, notre recherche a comporté cinq étapes. Dans la première étape, nous avons identifié les critères de Vincenti de la classification des connaissances de conception de l'ingénierie. Ces critères nous ont permis d'analyser, de façon structurée, divers modèles de maintenance du logiciel et de faire ressortir certaines insuffisances dans la définition de ces modèles. Ces critères de Vincenti sont considérés comme critères de comparaison sur lesquels nous nous sommes basés pour faire notre analyse comparative. Ensuite, nous avons analysé les modèles portant sur la maintenance du logiciel du projet SWEBOK, des travaux de Pigoski et de la norme ISO/IEC 14764 et en s'aidant des critères de Vincenti. Nous avons identifié dans chacun de ces modèles des forces et des faiblesses. La dernière étape a consisté à faire une étude comparative et donner des explications sur les différences identifiées. Dans cette étape, nous avons apporté des suggestions d'amélioration à ISO et à SWEBOK.

Notre recherche sur les modèles de maintenance nous a donné une ébauche de la réponse à certaines questions que nous avons posées. Par exemple, le manque d'accord dans la définition des terminologies. Essentiellement, nous avons proposé de définir, au niveau de la norme ISO/IEC 14764, un glossaire permettant de mettre en parallèle les termes ayant le même nom et qui ne définissent pas la même chose, et que les concepts créés doivent se baser non seulement sur les points de vue des experts mais aussi sur des observations systémiques dans un large éventail de contextes industriels et académiques. Dans le modèle de SWEBOK (version 0.7), nous avons proposé de bien identifier les critères portant sur les stratégies de maintenance.

Dans le domaine de la recherche, l'originalité de ce travail réside dans la considération du domaine de maintenance du logiciel dans un contexte de l'ingénierie. Nous avons utilisé les éléments de connaissances en ingénierie de Vincenti comme des critères permettant de juger de la maturité de maintenance pour aboutir à élaborer notre rapport de modèles améliorés.

Au niveau personnel, ce projet nous a permis d'acquérir des connaissances considérables dans le domaine du génie logiciel : la maturité du génie logiciel, le projet SWEBOK ainsi que l'utilité et la nature de la norme ISO.

Comme perspective d'avenir, nous préconisons de concevoir des catégories spécifiques qui s'appliquent directement aux connaissances de maintenance et qui sont inspirés des critères de Vincenti des connaissances de conception.

ANNEXE A

CADRE DE BASILI ADAPTÉ À LA RECHERCHE EXPLORATOIRE

par Abran et al.

1. Définition			
Motivation	**Portée**	**Objectif**	**Utilisateurs de recherche**
Aider à améliorer les modèles de maintenance du logiciel	Les modèles ISO/IEC 14764 et SWEBOK (version 0.7) sur la maintenance du logiciel	Proposer des suggestions d'amélioration des modèles de ISO/IEC 14764 et de SWEBOK sur la maintenance	Les chercheurs des principes fondamentaux de SWEBOK Les développeurs de normes en maintenance Les spécialistes dans le domaine

2. Planification		
Étapes du projet	**Intrants du projet**	**Livrables du projet**
Étude de la classification des critères de VINCENTI	Document de VINCENTI	***Livrables intermédiaires :*** Identification des critères de comparaison (de VINCENTI)
Analyse du modèle de maintenance de SWEBOK selon les critères identifiés	Critères identifiés Modèle de maintenance de SWEBOK	Modèle de maintenance de SWEBOK analysé forces et faiblesses
Analyse du modèle de maintenance de PIGOSKI selon les critères identifiés	Critères identifiés Modèle de maintenance de PIGOSKI	Modèle de maintenance de PIGOSKI analysé forces et faiblesses
Analyse du modèle de ISO/IEC 14764 selon les critères identifiés	Critères identifiés Modèle de ISO/IEC 14764	Modèle de ISO/IEC 14764 analysé forces et faiblesses
Étude comparative des divers modèles de maintenance de ISO/IEC 14764 et de SWEBOK (version 0.7)	ISO/IEC 14764, SWEBOK et les travaux de Pigoski	***Livrables finaux :*** Proposition du modèle de ISO/IEC 14764 amélioré Proposition du modèle de maintenance de SWEBOK (version 0.7) amélioré

3. Opération		
Analyse des documents	**Feedback des praticiens**	**Modèles proposés**
Découvrir les critères de classification de VINCENTI Vérifier les critères selon SWEBOK Vérifier les critères selon PIGOSKI Vérifier les critères selon ISO Identifier les forces et les faiblesses Rédaction du rapport final		Critères de VINCENTI : 1. concepts fondamentaux de conception 2. critères et spécifications 3. outils théoriques 4. données quantitatives 5. considérations pratiques 6. instruments de conception Proposition du modèle de ISO/IEC 14764 amélioré Proposition du modèle de SWEBOK amélioré Rapport final

4. Interprétation		
Contexte d'interprétation	**Extrapolation des résultats**	**Travaux futurs**
L'analyse des modèles de maintenance du logiciel selon les critères de Vincenti a permis d'identifier les forces et les faiblesses de ces modèles L'objectif d'amélioration des modèles de maintenance de ISO/IEC 14764 et de SWEBOK est conceptuel. Ne comprend aucun test d'implantation. Les améliorations proposées se basent sur la revue de la littérature. Cette recherche pourrait contribuer à mieux définir les modèles de maintenance du logiciel	L'étude des divers modèles de maintenance du logiciel dans un contexte de l'ingénierie a été réalisée. Il n'y a pas eu d'expérimentation et de vérification des résultats.	Rédaction d'articles et de documents pilotes de maintenance. Approfondissement de nouvelles pistes de recherche : Amélioration de la classification de Vincenti pour orienter les domaines du génie logiciel dans un contexte de l'ingénierie.

RÉFÉRENCES

Abran, A. et J.-P. Jacquet. 1999. « A Structured Analysis of the New ISO Standard on : Functional Size Measurement - Definition of Concepts (ISO/IEC 14143-1) ». *IEEE International Software Engineering Standards Symposium*, ISESS'99, Curitiba, Brazil.

Abran, A. et al. 1999. « A Risk Assessment Method and Grid for Software Engineering Measurement Programs », *Université du Québec à Montréal, Département d'informatique*, Montréal, Technical Report 99-03.

ANSI/IEEE STD 1061. 1998. « IEEE Standard for a Software Quality Metrics Methodology ». *IEEE Computer Society Press*, p. 3-13.

Arthur, L. J. 1988. *Software Evolution - The Software Maintenance Challenge*. John Wiley & Sons.

Basili, V.R. 1985. « Quantitative Evaluation of Software Methodology ». *Proceedings of the First Pan-Pacific Computer Conference*.

Basili, Victor R., David H. Hutchens et Richard W. Selby. 1986. «Experimentation in software engineering ». *IEEE Transactions on software engineering*, Vol. SE-12, no 7, July, p. 733-743.

Bohem, B. W. 1981. *Software Engineering Economics*. Prentice-Hall, p. 534-553.

Bourque, P., R. Dupuis; A. Abran, J.W. Moore et L.L. Tripp, 1999. «Le guide du corpus des connaissances en génie logiciel» in *Génie Logiciel*. Traduit et reproduit de « The Guide to the Software Engineering Body of Knowledge », *IEEE Software*, Vol. 16, no 6, p. 35-44., Vol. 51, p. 3-12.

Bouthat, Chantal. 1993. *Guide de présentation des mémoires et thèses*. Université du Québec à Montréal.

Card, D.N. et R.L. Glass. 1990. *Measuring Software Design Quality. Englewood Cliffs*, NJ : Prentice-Hall.

Dekleva, S. M. 1992. « Delphi Study of Software Maintenance Problems ». *Proceedings of the International Conference on Software Maintenance*, p. 10-17.

Dorfman, M. et R. H., Thayer. 1997. « Software Engineering ». *IEEE computer Society Press*.

Faulkner, Wendy. 1994. « Conceptualising Knowledge Used in Innovation : A Second Look at the Science-Technology Distinction and Industrial Innovation ». *Science, Technology, & Human Values*, Vol. 19, no. 4, Autumn, p. 425-458.

Grady, R.B. et Caswell, D. L. 1987. *Software Metrics : Establishing a company-wide Program.* Prentice-Hall, Chapter 1.

Grady, R.B. 1992. *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs, NJ : Prentice-Hall.

Grady, R. B. 1994. « Successfully Applying Software Metrics ». *IEEE Computer Society Press.*

IEEE729. 1983. « IEEE Standard Glossary of Software Engineering Terminology ».

IEEE610.12. 1990. « IEEE Standard Glossary of Software Engineering Terminology ». *IEEE STD 610.2*.

IEEE1219. 1998. « Standard for Software Maintenance ». *IEEE STD 1219.*

IEEE Computer Society. 1998. *Conference on Software Maintenance*. Computer Press. Los Alamitos.

IEEE Computer Society. 1999. *International Conference on Software Maintenance*. Computer Press. Los Alamitos.

ISO/IEC 14471 « Information technology—Guidelines for the Adoption of CASE tools ». *ISO/IEC DTR 14471*.

ISO/IEC 9126. 1999. « ISO/IEC 9126 - Software Product Quality ».

ISO 8402. 1994. « Quality Management and Quality Assurance Vocabulary ». Geneva, Switzerland.

ISO/IEC 12207. 1995. « Information Technology-Software Life Cycle Processes ».

ISO/IEC 14764. 2000. « Software Engineering-Software Maintenance ». JTC 1/SC 7 N°2043.

Jones, C. 1994. *Assessment and Control of Software Risks. Englewood Cliffs.* NJ : Prentice-Hall.

Lehman, M. et L.A. Belady. 1986. *Program Evolution – Processes of Software Change.* Academic Press Inc. (London) Ltd.

Lientz, B.P. et E.B. Swanson. 1981. « Problems in Application Software Maintenance ». *Communications of the ACM*, Vol. 24, no 11, p. 763-769.

Martin, J. et C. McClure. 1983. *Software Maintenance : The Problem and its Solutions*. Englewood Cliffs, NJ : Prentice-Hall.

Parikh, G. et N. Zvegintzov. 1993. «Tutorial on Software Maintenance ». *CA :* IEEE Computer Society Press, Los Alamitos.

Pfleeger, Shari Lawrence. 1998. *Software Engineering : Theory and Practice*. Prentice-Hall, Inc., Upper Saddle River.

Pigoski, Thomas M. 1997. *Practical Software Maintenance : Best Practices for Managing your Software Investment*. John Wiley & Sons, Inc., Canada.

Pigoski, Thomas M. 1999. « SWEBOK Knowledge Area Description for Software Evolution and Maintenance (Draft version 0.7) ». *http://www.swebok.org*.

Pressman, Roger S. 1997. *Software Engineering : A practitioners approach*. Fourth edition McGraw-Hill.

Rakotomala, José. 1998. « Élaboration d'un cadre d'évaluation des systèmes d'évolution de schéma dans les bases de données à objets ». *Mémoire de maîtrise en informatique de gestion*, Montréal, Université du Québec à Montréal.

Sharpe, S., D. A. Haworth, et D. Hale. 1991. «Characteristics of Empirical Software Maintenance Studies : 1980-1989 ». *Journal of Software Maintenance Research and Practice*, Vol. 3, p. 1-5.

Smith, Dennis D. 1999. *Designing Maintainable Software*, Springer c1999. New York.

Somerville, I. 1996. *Software Engineering*. McGraw-Hill, Fifth edition, p. 121-124.

Somerville, I. 2001. *Software Engineering*. Addision-Wesley. Sixth Edition.

Stark, G. E., L. C. Kern et C.V. Vowell. 1994. « A Software Metric Set for Program Maintenance Management ». *Journal of Systems and Software*.

Swanson, Burton E. et Cynthia M. Beath. 1989. *Maintaining information systems in organisations*. John Wiley & Sons Ltd, Chichester. New York.

Swebok. 2000. « Guide to the Software Engineering Body of Knowledge ». *http://www.swebok.org/documents/stoneman07/Guide_to_the_SWEBOK-Stone_Man_Version_07.PDF*

Takang, Armestrong A. et Penny A. Grubb. 1997. *Software Maintenance : Concepts and Practice*. International Thomson Computer Press. London.

Vincenti, Walter G. 1990. *What Engineers Know and How they Know It : Analytical Studies from Aeronautical History*. The Johns Hopkins University Press.

Von Mayrhauser, T.E. 1990. *Software Engineering-Methods and Management*. San Diego, CA : Academic Press, Inc.

Zitouni, M. 1996. « Élaboration d'un outil d'amélioration du processus de maintenance des logiciels : étude exploratoire ». *Rapport d'activité de synthèse de la Maîtrise en informatique de gestion*, Montréal, Université du Québec à Montréal.

Zvegintzov, N. 1994. *Software Management Technology*. Reference guide Software Maintenance News Inc., Edition.