UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ANALYSE COMPARATIVE DES MODÈLES DE MAINTENANCE DU LOGICIEL ENTRE SWEBOK, ISO/IEC 14764 ET LA LITTÉRATURE

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

PAR
ASMA SELLAMI

JUIN 2000

Table des matières

INTRODUCTION

Au cours des dernières années, les effets de maintenance du logiciel ne sont pas limités à un individu ou à une compagnie. À travers l'échange d'information, les sociétés du monde sont devenues globalement intégrées. Les réseaux d'ordinateurs et les équipements des télécommunications interconnectent le monde et permettent aux communautés distantes d'interagir et de travailler ensemble. Ainsi, plus les compagnies du monde avancent et deviennent interdépendantes, plus elles deviennent dépendantes des systèmes qui génèrent et distribuent l'information. Un tel système dépend de la fonctionnalité, de la flexibilité et des opérations de modification des produits logiciels.

La maintenance de logiciels est la modification d'un produit logiciel après sa livraison. C'est une des fonctions la plus importante dans le cycle de vie de logiciel. De plus, les coûts de 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 de logiciel [Pigoski, 1997].

Dans ce document, nous nous intéressons aux modèles de maintenance de logiciels. Ces derniers, représentent des plans pour la maintenance et nécessitent en particulier d'être exécutés, évalués et ajustés à la porté d'un produit logiciel donné. Un des objectifs de ces modèles est de fournir un guide pour les mainteneurs.

Plusieurs modèles de maintenance ont été proposés dans la littérature. Selon la norme ISO/IEC 14764 [ISO 14764], ces modèles sont conçus de point de vue de gestion. D'autre part, dans le projet SWEBOK [SWEBOK, 2000], c'est l'aspect connaissance qui est mis en relief.

Les budgets accordés à la fonction de maintenance ne cesse de croître. Elle représente entre 50 et 70% des coûts totaux du logiciel selon différentes sources [Arthur, 1988 ; Swanson et Beath, 1989 ; Sharpe et al., 1991]. Cependant, malgré cette augmentation la maintenance continue à être relativement négligée dans un contexte du génie logiciel.

Dans cette étude exploratoire, on analyse les modèles de maintenance du logiciel dans un contexte de l'ingénierie, et ceci en se basant sur la classification de Vincenti [Vincenti, 1990].

L'objectif principal de cette recherche, défini à partir du cadre de Basili [Basili, 1986] modifié par Abran [Abran et al., 1997], est de proposer des suggestions d'amélioration des modèles ou des documents de synthèse sur la maintenance et le génie en général, particulièrement, ceux de la norme ISO/IEC 14764 [ISO 14764] et du projet SWEBOK [SWEBOK, 2000].

On vise ainsi d'explorer, dans un contexte de l'ingénierie, un nouveau rapport comportant aussi bien le modèle du projet SWEBOK amélioré que celui de l'ISO amélioré sur la maintenance du logiciel. Ces modèles seront basés sur des éléments conceptuels liés à la classification de Vincenti.

L'application de cette classification est orientée vers l'identification et la clarification des sous-domaines de connaissance liés à la maintenance du logiciel de point de vue de l'ingénierie. Comme pour tout travail mature, les éléments de la classification ou la catégorisation de chaque connaissance seront présents.

Dans le cadre de cette recherche exploratoire, un procédés exploratoire fournit une base pour l'avancement utile en connaissance et en compréhension. Puisque le domaine de maintenance du logiciel est encore immature, il est certainement candidat pour les méthodes d'analyse exploratoire. Ces méthodes seront accomplies pour aider à mieux comprendre, contrôler et améliorer les documents de synthèse liés à la maintenance du logiciel.

Dans le premier chapitre de ce mémoire, nous présenterons la problématique, ainsi que la méthode qui guidera la mise en œuvre de ce projet de recherche. Cette démarche est basée sur le cadre de [Basili et al., 1986] modifié par le Laboratoire de recherche en gestion des logiciels de l'UQAM, par [Abran et al., 1997].

Nous formulons ainsi, la motivation, l'objet, l'objectif de la recherche, les divers utilisateurs à qui s'adresse ce mémoire et la planification incluant les étapes du projet, les intrants et les livrables ainsi que les limites de recherche.

Les chapitres qui suivent portent chacun sur une revue de la littérature des différents textes de :

1. SWEBOK : swebok knowledge area description for software maintenance (version 0.7),
2. Pigoski : Practical software maintenance, best practices for managing your software investment (1997),
3. ISO/IEC JTC1/SC7 N2043 : FDIS 14764, Software Engineering - software maintenance (1998),

4. Autres littératures de Swanson : Maintaining information systems in organisations (1989).

On décrit et on analyse chacun de ces travaux en se basant sur les critères de Vincenti identifiés afin de dégager les forces et les faiblesses. Ces étapes correspondent à la phase opération du cadre de Basili adapté pour une étude exploratoire.

Dans le dernier chapitre, nous présenterons l'analyse des résultats, ce qui constitue la dernière phase du cadre modifié de [Basili et al., 1986], et qui porte sur l'interprétation des résultats : le contexte d'interprétation incluant une synthèse de SWEBOK, de l'ISO et de la littérature pour identifier les désaccords qui peuvent exister entre les différents travaux, l'extrapolation des résultats de cette recherche permettant de préparer des recommandations des modèles de maintenance et les travaux futurs à entreprendre à partir des résultats trouvés.

PRÉSENTATION DU TRAVAIL

1.1 PROBLÉMATIQUE DE RECHERCHE

La maintenance de logiciels est définie par le standard [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. Selon des littératures récentes, cette fonction doit être réalisée au moment de la prise de décision de développement d'un système logiciel. Cependant, elle devient de plus en plus dispendieuse entre toutes les différentes phases du cycle de vie de logiciel.

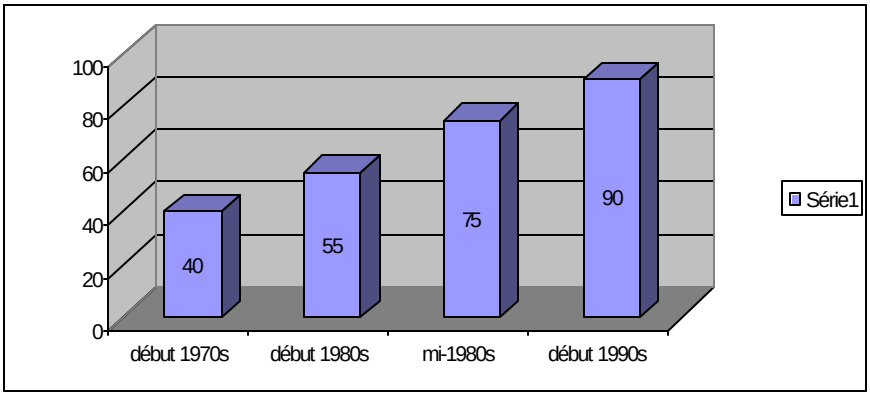
D'après une étude réalisée par [Swanson et Beath, 1989], le budget accordé à la maintenance est relativement élevé, il est de l'ordre de 50 à 70% des coûts totaux du logiciel.

Selon une étude effectuée auprès des entreprises américaines, 30 milliards US $ sont dépensés annuellement pour la maintenance du logiciel dans les années 1990, et 95% des coûts du cycle de vie d'un logiciel est affectée à la maintenance [Pigoski, 1997].

La figure suivante montre l'évolution des coûts de maintenance de logiciels :

Figure 1.1 : Coûts alloués à la maintenance dans le cycle de vie de logiciel (Source Pigoski, 1997)

À Partir de cette figure, on peut constater que les coûts consacrés à l'activité de maintenance de logiciels augmentent malgré les efforts réalisés dans le domaine du génie logiciel couvrant les différentes activités du cycle de vie de logiciel.



Parmi toutes les explications que l'on peut donner à ces coûts élevés, on peut citer par exemple l'inexistence de la documentation et que les mainteneurs consomment beaucoup de temps dans la compréhension du code.

Dans la littérature, plusieurs auteurs ont énoncé plusieurs modèles de maintenance de logiciels et on retrouve ceux qui s'entendent sur un modèle global. Néanmoins, le problème réside dans le fait qu'il y a une certaine défaillance entre ce qui se trouve dans la littérature et ce qui est appliquée dans la pratique. En particulier, les modèles ou les documents de synthèse sur la maintenance de la norme ISO sont développés par discussion d'experts et non vérifiés par autres que les participants. 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. Malgré des systèmes sont maintenus annuellement, relativement peu d'études qui portent sur la maintenance que ce soit de point de vue pratique ou académique. Ce manque de documentation contribue à une confusion dans la compréhension des gens sur la maintenance.

En plus, plusieurs définitions sont accordées à la maintenance de logiciels :

- La maintenance des logiciels est la modification d'un produit logiciel après sa livraison pour corriger les erreurs, améliorer la performance ou adapter le produit à un environnement modifié. Il s'agit de mettre en relief les activités de maintenance préalable à la livraison d'un produit logiciel [IEEE 1219],
- La maintenance signifie l'évolution des logiciels [Sommerville, 1996],
- La maintenance est décrite comme étant le processus d'un produit logiciel sous la modification du code et les documents associés suite à un problème ou un besoin d'amélioration. Il s'agit de modifier le produit logiciel existant tout en gardant son intégrité [ISO 12207],
- La maintenance des logiciels est définie en même terme que ISO/IEC 12207 [ISO 12207] et l'accent se fait sur les aspects pré-livraison de maintenance comme par exemple : la planification [ISO 14764],
- La maintenance des logiciels est décrite par les praticiens comme étant la totalité des activités requise pour fournir un support efficace à un système logiciel. Ces activités sont exécutées aussi bien durant les étapes pré-livraison qu'après la livraison [SWEBOK, 2000].

D'autre part, selon la norme ISO/IEC 14764 [ISO 14764], la qualité de 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 doivent avoir un programme de qualité de logiciel répondant aux caractéristiques de qualité de la norme ISO/IEC 9126 [ISO 9126]. Un processus doit être implanté afin d'identifier, définir, sélectionner, appliquer, valider et améliorer la

mesure de logiciel pour la maintenance de logiciels. Comme partie de mesure de logiciels, le mainteneur 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 de logiciels et afin d'obtenir une meilleure compréhension de la répartition des coûts de maintenance. Les données de mesure empirique doivent être collectées pour aider à l'estimation des coûts du cycle de vie.

La maintenance de logiciels présente différents problèmes pour le génie logiciel. Selon Pfleeger [Pfleeger, 1998], les problèmes liés à la maintenance sont d'ordre technique et de gestion :

1.1.1 Problèmes techniques

Il y a plusieurs problèmes de personnel et organisationnels qui rendent difficile la fonction de maintenance. Le personnel doit agir comme intermédiaire entre le problème et sa solution pour s'assurer que toute modification répond aux besoins.

1.1.1.1 Compréhension limitée

Plusieurs études montrent que 40 à 60% de l'effort de maintenance est dévoué à la compréhension du logiciel à modifier. Ainsi, le thème de compréhension de programme est extrêmement important pour les mainteneurs. Il est souvent difficile de suivre l'évolution d'un logiciel selon ses versions, les modifications ne sont pas documentées et les développeurs ne s'entendent pas pour expliquer le code. À cet effet, les mainteneurs ont une compréhension limitée de logiciel et doivent fournir un effort pour étudier individuellement le logiciel.

1.1.1.2 Test

Tester peut être un problème quand il n'y a pas assez de temps pour tester. Par exemple, le système de réservations aériennes doit être toujours disponible. Il est souvent 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 l'interception d'un patient, il est presque impossible d'arrêter le système afin de le tester.

Dans ces cas de figure, les tests sont souvent exécutés sur des systèmes dupliqués; ensuite les changements sont transférés aux systèmes de production.

En plus des problèmes de disponibilité de temps, il n'y a pas un test approprié des données pour tester les changements réalisés. Comme par exemple, les données de test de prévision de tremblement de terre ne sont pas généralement précises.

Ainsi, il n'est pas toujours facile pour les testeurs de prévoir et de préparer les effets de conception ou de modifications de code. Cette incertitude existe spécialement quand les différents membres de l'équipe de maintenance travaillent simultanément sur différents problèmes. Si, en même temps, deux mainteneurs dont l'un apporte des modifications sur un composant pour résoudre un problème de débordement par exemple et l'autre apporte une modification sur la même composante pour résoudre un problème d'interface, ils peuvent créer de nouveaux défauts dans le système.

1.1.1.3 Analyse de l'impact

Le logiciel et l'organisation doivent tous effectuer des analyses de l'impact. Des habiletés critiques et des processus sont nécessaires dans ce domaine. Ainsi, l'analyse de l'impact est nécessaire pour minimiser le risque.

Une revue après-opération peut être essentielle pour évaluer l'impact de changement à un nouvel environnement.

1.1.1.4 Maintenabilité

La norme IEEE [IEEE 610.12] définie la maintenabilité comme la facilité selon laquelle un logiciel peut être maintenu, amélioré, adapté ou corrigé pour satisfaire les besoins spécifiés. Les caractéristiques de maintenabilité doivent être incorporées dans l'effort de développement de logiciel pour réduire les coûts du cycle de vie. Si cette condition est vérifiée, la qualité de maintenance de code peut s'améliorer. Un des problèmes de maintenance est la maintenanbilité, car cette dernière n'est pas souvent incorporée dans le processus de développement de logiciel, la documentation est incomplète voir inexistante et la compréhension des programmes difficiles. Les moyens d'améliorer la maintenanbilité sont la définition des standards pour le code, des standards pour la documentation et des outils de test de standard dans la phase de développement du cycle de vie [Dorfman et Thayer, 1997].

1.1.2 Problèmes de gestion

1.1.2.1 Les priorités de gestion

L'équipe de maintenance prend en compte les désirs de gestion des clients versus les besoins du système. Souvent, les priorités de gestion ont plus d'importance que les priorités techniques ; Parfois, les gestionnaires considèrent la maintenance et l'amélioration comme étant plus importante que le développement de nouvelles applications. En d'autres termes, les compagnies se concentrent sur les affaires plutôt que d'investir dans de nouvelles alternatives. Cependant, comme les gestionnaires encouragent les mainteneurs à maintenir l'ancien système, les utilisateurs doivent s'adapter à de nouvelles fonctions dans le système. D'autre part, la précipitation d'avoir le produit sur le marché peut encourager aussi bien les développeurs que les mainteneurs à implanter un changement rapide, inélégant, pauvrement testé plutôt que de prendre suffisamment le temps pour suivre les bonnes pratiques du génie logiciel. Le résultat est un produit lancé qui est difficile à comprendre et à maintenir ultérieurement.

1.1.2.2 Moral

Des études réalisées par [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 sont qualifiés non seulement à écrire le code mais aussi à travailler avec les utilisateurs dans l'anticipation des changements et dans les limites. Beaucoup d'habilités et de persévérances sont nécessaires pour comprendre les travaux internes d'un gros système et pour modifier la structure du système, le code et la documentation.

Certains membres de l'équipe de maintenance remplissent leur fonction à tour de rôle dans plusieurs projets pour faire le maximum dans un intervalle de temps prédéterminé. Cette rotation peut engendrer des problèmes de conflits de priorités. Durant la maintenance, 8% des problèmes résultent de l'appartenance des mainteneurs dans plusieurs directions à la fois, et ainsi ils deviennent incapables de bien se concentrer sur un problème donné.

1.1.2.3 Alignement avec des problèmes organisationnels

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

D'autre part, plusieurs documents de synthèse sur la maintenance sont identifiés par la norme ISO/IEC 14764 [ISO 14764], le projet SWEBOK et la littérature. Ces documents sont développés à la discussion avec des experts mais ne sont vérifiés que par les participants.

1.2 MÉTHODE DE LA RECHERCHE

La méthode à suivre dans cette recherche 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 : les principes fondamentaux du génie logiciel et la mesure de la taille fonctionnelle des logiciels.

Ce cadre a été élaboré initialement aux cas des études de type expérimentale pour fournir un protocole de recherche simple, bien structuré et facile à utiliser. Il comporte essentiellement quatre phases de déroulement d'une expérience : 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 par Abran [Abran et al., 1997] principalement pour documenter et réaliser des projets de recherche exploratoire.

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.

1- Définition			
Motivation	Objet	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

Tableau 1.2 : Cadre de Basili adapté à la recherche exploratoire par [Abran et al.]

La phase définition comporte les étapes suivantes : la motivation, l'objet, l'objectif et les utilisateurs de recherche. Les étapes du projet, les intrants et les livrables du projet sont inclus dans la phase de planification. La phase opération est composée des étapes suivantes : l'analyse des documents, le feed-back des praticiens et le modèle proposé. Finalement, la dernière phase d'interprétation est formée des étapes suivantes : le contexte d'interprétation, l'extrapolation des résultats et les travaux futurs.

1.3 DÉFINITION DU PROJET DE RECHERCHE

Dans cette section on présente le contenu de la première phase de Basili modifié permettant de définir les objectifs et les limites de cette recherche.

1.3.1 Motivation, objet et objectif de l'étude

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

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

L'objectif principal de notre recherche consiste à proposer des suggestions d'amélioration aux documents de synthèse des connaissances sur la maintenance du logiciel et le génie en général. Pour cela, nous avons à effectuer une analyse comparative des domaines de connaissance entre SWEBOK, la norme ISO/IEC 14764 [ISO 14764] et la littérature et en s'aidant à partir des critères de Vincenti.

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

On pose ainsi la question de recherche suivante :
Existe-t-il des divergences entre SWEBOK, ISO/IEC 14764 et la littérature sur la façon de considérer la maintenance des logiciels en se basant sur les critères de Vincenti?

Cette question peut être subdivisée en quatre sous-questions :

1- Est-ce que la littérature supporte le contenu de la norme ISO/ IEC 14764?
2- Est-ce que les connaissances généralement acceptées par SWEBOK sont compatibles avec ce qui se trouve dans les normes?
3- Est-ce que la littérature supporte le contenu de SWEBOK?
4- Si des faiblesses apparaissent dans les connaissances sur la maintenance telles qu'évaluées à l'aide des catégories de Vincenti, trouve-t-on dans la littérature de quoi combler ce vide?

1.3.2 Utilisateurs de recherche

Notre projet de recherche préoccupe de nombreuses personnes dont les activités sont liées d'une façon ou d'une autre à la maintenance du logiciel :

- Les chercheurs des principes fondamentaux du projet SWEBOK, qui pourrons profiter d'une liste de principes importants sur le domaine de connaissance lié à la maintenance.
- Les développeurs de normes en maintenance, qui pourront disposer d'un nouveau rapport comportant des modèles de maintenance améliorés.
- 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, nous avons identifié les étapes qui suivent :

1. Nous procédons à une étude de la classification de Vincenti pour identifier les critères de comparaison. Cette classification comporte :

   - Les concepts fondamentaux de conception,
   - Les critères et les spécifications,
   - Les outils théoriques,
   - Les données quantitatives,
   - Les considérations pratiques,
   - Les instruments de conception.

2. Analyse des travaux selon les critères de Vincenti identifiés pour dégager les forces et les faiblesses :

   - Texte de Pigoski (1997),
   - SWEBOK : swebok knowledge area description for software maintenance (Version 0.7),
   - ISO : ISO/IEC 14764,
   - Autres littératures : Texte de Swanson (1989).

3. Les étapes d'analyse à suivre sont :

   - Classification de chaque travail selon les critères identifiés,
   - Vérification si chacun de ces travaux répond à tous les critères identifiés,
   - Identification des forces et des faiblesses.

4. Synthèse de SWEBOK, de la norme ISO/IEC 14764 et de la littérature.

   - Identification des désaccords qui peuvent exister entre les différents travaux.

5. Dans la dernière étape de la recherche, nous procédons à une préparation des recommandations des modèles ou des documents de synthèse des connaissances de maintenance de logiciels.

1.5 LIMITE DE LA RECHERCHE

La recherche n'étudiera que les modèles de maintenance de logiciels 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 à ces modèles.

Cette étude est basée sur une revue de la littérature. Cette revue aura pour objectif de nous donner une meilleure compréhension de notre sujet et de nous permettre de présenter des arguments solides.

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

- Designing maintainable software [Denis D. Smith, 1999],
- Software engineering : theory and practice [Pleeger Shari Lawrence, 1998],
- Software maintenance [Armstrong A. Takang et Penny A. Grubb, 1996],
- Maintaining information systems in organisations [E. Burton Swanson et Cynthia Mathis Beath, 1989].
- Software Engineering A practitioner's approach [Royer S. Pressman, 1997].

Il est clair que cette recherche n'implique pas la gestion de 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, selon la classification de Vincenti, le contenu d'un domaine de connaissance en ingénierie, particulièrement la maintenance de logiciels du point de vue de l'ingénierie et non pas du point de vue strictement de codification.

1.6 CLASSIFICATION DE VINCENTI

1.6.1 Pourquoi a-t-on choisi la classification de Vincenti?

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

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 au cours des années. Son livre présente en premier lieu un certain nombre d'études de cas historiques dans les années 1908 et 1953 et appartenant à une variété de problèmes de conception des avions : la conception des ailes, le développement de la théorie d'analyse contrôle-volume, etc.

Un chapitre clé intitulé « The Anatomy of Engineering Design Kowledge » montre ce que Vincenti propose comme structure des connaissances du génie en général et ceci en présentant une tentative de catégorisation ou de classification de chaque connais sance basée essentiellement sur des études de cas et de l'expérience propre de l'auteur sur le génie. M. Vincenti admet qu'une telle généralisation peut être un risque puisqu'elle provient du domaine aéronautique, néanmoins, il croit que cette classification peut être universellement applicable (Réf. Vincenti, page 200). Les six catégories des connaissances du génie qu'il propose sont les suivants :

1.6.1.1 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 fait, M. Vincenti énonce que ces concepts peuvent seulement exister d'une manière implicite dans l'esprit du concepteur. Ils sont donnés pour un projet, même s'il n'est pas spécifié. Ils sont absorbés par les ingénieurs au cours de leurs croissances, même 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.

1.6.1.2 Les critères et les spécifications

Ce type de connaissance permet à l'ingénieur ou au concepteur de :

- identifier les besoins spécifiques en terme de matériel,
- traduire les buts qualitatifs d'un système (le général) en des buts quantitatifs cachés dans des termes techniques concrets (le spécifique).

Pour accomplir ces tâches, les concepteurs doivent avoir des connaissances sur les critères techniques appropriés d'un système et de son utilisation et doivent attaquer en quelque sorte les valeurs

numériques spécifiques ou les limites de certains critères techniques. Sans ces spécifications techniques, il est presque difficile, pour un concepteur, de fournir les détails et les dimensions nécessaires aux développeurs.

Cependant, Vincenti indique que la connaissance clé est la sélection d'un ensemble appropriés de critères. En effet, les critères de conception varient largement selon la perception. 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.

Quand les circonstances sont suffisamment générales, les spécifications des valeurs ou des limites peuvent être appliquées à travers la technologie. Ces spécifications se prolifèrent dans le cas où l'utilité générale serait impliquée. Comme par exemple, les spécifications universelles qui deviennent une partie des connaissances : Comment sont réalisées les choses dans l'ingénierie ?

La détermination des critères essentiels dépend des outils théoriques, des données quantitatives et du jugement pragmatique dans les critères suivants.

1.6.1.3 Les outils théoriques

De tels outils s'étendent des modèles mathématiques et des théories nécessaires pour l'analyse et la conception quantitatives, ainsi que des concepts intellectuels nécessaires pour la conceptualisation qualitative et le raisonnement (« des concepts pour penser à la conception »). De tels concepts et outils incluent la connaissance scientifique générale appelés par Vincenti « des théories phénoménologiques », des techniques *ad hoc*, approximation ou des théories utiles seulement pour le calcul du génie.
Les outils théoriques nécessaires pour 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. Ces concepts sont employés non seulement dans l'analyse quantitative mais également dans la conception qualitative.

1.6.1.4 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 perspectives (comment les choses devraient être?, par exemple, facteur de sûreté), est d'aider à déterminer les détails des systèmes à concevoir.

1.6.1.5 Les considérations pratiques

Ce sont des connaissances basées empiriquement sur des expériences ou des conventions. Généralement, une telle connaissance n'est pas formellement codifiée. Elle est souvent représentée par des règles de « thumb ». Selon Vincenti, quand ce type de connaissance devient formellement enregistrer et codifier, il devient souvent une partie d'une autre catégorie de connaissance.

1.6.1.6 Les instruments de conception

Ce sont les approches 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 de « Comment » accomplir les tâches.

Il est à noter que l'identification de chaque thème de connaissance en se basant sur la catégorisation de Vincenti n'est pas une tâche facile. En effet, comme Vincenti le déclare, ces catégories ne sont pas mutuellement exclusives car, les frontières entre ces catégories sont relativement confuses. Cependant, M. Tremblay, auteur du document Jump-Start pour le domaine de connaissance de la construction des logiciels et spécialiste dans le domaine de conception des logiciels, a fourni une classification dans les thèmes des domaines de connaissances en utilisant ces catégories.

Une catégorisation concernant le domaine des connaissances de maintenance des logiciels basée sur la classification de Vincenti a été produite par T. Pigoski [Pigoski, 1997] dans le projet SWEBOK (version 0.7). Ce qui fait l'objet du chapitre suivant

Annexe A
Cadre de Basili adapté à la recherche exploratoire par Abran et al.

1- Définition			
Motivation	**Objet**	**Objectif**	**Utilisateurs de recherche**
Aider à améliorer les modèles de maintenance du logiciel	Les modèles SWEBOK et ISO de maintenance du logiciel	Proposer des suggestions d'amélioration des modèles de maintenance du logiciel	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 de VINCENTI	VINCENTI	***Livrables intermédiaires :*** Identification des critères
Analyse de SWEBOK	Critères identifiés Texte de SWEBOK	Texte de SWEBOK analysé forces et faiblesses
Analyse du texte de PIGOSKI selon les critères	Critères identifiés Texte de PIGOSKI	Texte de PIGOSKI analysé forces et faiblesses
Analyse de l'ISO	Critères identifiés Texte de l'ISO	Texte de l'ISO analysé forces et faiblesses
Analyse autres littératures	Critères identifiés Autres littératures (SWANSON)	littératures analysées forces et faiblesses
Synthèse : Analyse comparative	PIGOSKI, SWEBOK, ISO et la littérature	***Livrables finaux :*** Proposition de modèle de SWEBOK amélioré Proposition de modèle de l'ISO amélioré

3- Opération		
Analyse des documents	**Feedback des praticiens**	**Modèle proposé**
Découvrir les critères de classification de VINCENTI Vérifier les critères selon SWEBOK Vérifier les critères selon PIGOSKI Vérifier les critères selon ISO Identifier les différences Faire une synthèse		Classification de VINCENTI : Les concepts fondamentaux de conception Les critères et les spécifications Les outils théoriques Les données quantitatives Les considérations pratiques Les instruments de conception Proposition du modèle SWEBOK amélioré Proposition du modèle ISO amélioré

4- Interprétation		
Contexte d'interprétation	**Extrapolation des résultats**	**Travaux futurs**

RÉFERENCE

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 (Juillet), p. 733-743.

Bourque et al. 1999. *Génie logiciel*, Vol 51, p. 3-12.

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

Dorfman, M. et Thayer, R. H. 1997. *Software Engineering IEEE computer Society Press*. G. E. Stark, L. C. Kern et C. V. Vowell. 1994 «A Software Metric Set for Program Maintenance Management» *Journal of systems and software*.

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

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

IEEE Computer Society. 1998. *Conference on Software Maintenance*.

IEEE Computer Society 1999 *International Conference on Software Maintenance* .

IEEE (*Standard for a quality metrics methodology*, ANSI/IEEE 1061- 1998)

ISO/IEC JTC 1/SC 7 N°2043, 1998 Software Engineering-Software Maintenance, December.

Lientz B.P. & E.B. Swanson (1981). « Problems in application software maintenance » Communications of the ACM, 24(11) : 763-769.

Parikh, G., & N. Zvegintzov (1993). Tutorial on software maintenance. Los Alamitos, CA : IEEE Computer Society Press.

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 practitioner's approach*, (quatrième édition) 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.

Smith, Denis D. 1999. *Designing maintainable Software*, Springer c1999. New York.

Swanson, Burton E. Bath, Mathis Cynthia. 1989. Maintaining information systems in organisations, John Wiley & Sons Ltd, Chichester. New York. Brisbane. Toronto. Singapore.

Takang, Armestrong A. et Grubb, Penny A. 1996. *Software maintenance : concepts and practice*, A. A. Takang and P. A. Grubb, International Thompson 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.

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.