Université du Québec
École de technologie supérieure

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS MGL 804 RÉALISATION ET MAINTENANCE DE LOGICIELS

TRAVAIL DE SESSION INDIVIDUEL

PAUL-OLIVIER TRUDEAU
TRUP19018209

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur superviseur

ALAIN APRIL

MONTRÉAL, 28 MARS 2012
HIVER 2012

TRAVAIL DE SESSION INDIVIDUEL

PAUL-OLIVIER TRUDEAU
TRUP19018209

RÉSUMÉ

Ce travail se veut un ensemble d'amélioration qui pourra être intégrée à la prochaine révision de la norme ISO/IEC 14764. Il s'agit en quelque sorte d'une critique de cette norme. Chaque suggestion est accompagnée d'une description de la problématique et est supportée par des références à des articles de la littérature du monde du génie logiciel.

Tout d'abord, le travail fait une critique des définitions proposées dans la norme. Les processus que propose la norme sont aussi regardés, plus particulièrement au niveau de la mesure ainsi que l'impact des méthodes agiles sur ces processus. Une critique du chapitre des considérations d'exécution est aussi proposée, plus particulièrement sur les divers environnements de la maintenance, sur les ententes de services et les outils. Finalement, une révision des stratégies de maintenance est proposée afin d'améliorer la norme en matière d'estimation de coûts et de prise en charge d'application spécifique.

Le travail contient aussi, en annexe, la liste complète des lacunes trouvées ainsi que les modifications proposées selon le gabarit de l'ISO/IEC/JTC1.

Réalisé dans le cadre du cours Réalisation et maintenance de logiciels (MGL804) à l'ÉTS au cours de la session d'Hiver 2002 avec le professeur Alain April.

TABLE DES MATIÈRES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

COSMIC-FFP:
Common Software Measurement International Consortium - Full Function Points

ÉTS:
École de technologie supérieure

FDIS:
Final Draft International Standard

IEC:
International Electrotechnical Commission

ISO:
International Organization for Standardization

JTC:
Joint Technical Committee

MGL:
Maîtrise en génie logiciel

SaaS:
Software as a service

INTRODUCTION

La norme ISO/IEC 14764 sert à décrire les processus pour la gestion et l'exécution des activités de maintenances logiciels. Cette norme a originalement été publiée en 1999, puis révisée en 2006. Mais les pratiques et processus qui y sont décrits sont-ils encore valides? Requièrent-ils une mise à jour? Certaines nouvelles pratiques nécessitent peut-être que la norme soit mise à jour?

Bien que la norme ISO/IEC 14764 n'est pas un modèle de maintenance logiciel, elle représente l'ensemble des pratiques à mettre à place afin de faire correctement la maintenance logicielle. Elle contient notamment un glossaire des définitions et des termes à utiliser dans le cadre de la maintenance logiciel, l'ensemble des processus à suivre, des stratégies de maintenances et un ensemble de considérations dont on doit tenir compte dans le contexte de la maintenance logiciel.

Plusieurs avancements technologiques et modifications des pratiques en génie logiciel ont eu lieu depuis 2006. Que ce soit la montée en popularité des méthodes agiles ou l'avènement des technologies web (SaaS, informatique en nuage, services web, etc..), le paysage et l'environnement ont considérablement évolué. Une norme se doit d'être vivante et d'être une représentation de ce qui se fait dans l'industrie.

Le présent travail présenté des améliorations possibles à la norme. Ce document se veut une critique de la norme ISO/IEC 14764 et proposera une liste d'amélioration qui pourrait être intégré la norme lors d'une prochaine révision.

Les modifications que je propose concernent les sujets suivants :

- Mise à jour des définitions
- Modifications des processus
- Ajouts de nouvelles considérations d'exécution
- Mise à jour des stratégies de maintenance

Ce travail a été réalisé dans le cadre du cours Réalisation et maintenance de logiciels (MGL804) à l'École de technologie supérieure, lors de la session d'hiver 2012 sous la direction du professeur Alain April.

CHAPITRE 1 : MISE À JOUR DES DÉFINITIONS

1.1 Maintenabilité

La majorité des normes produites par l'ISO ou par tout autre organisme de normalisation contient une section listant l'ensemble des définitions qui seront utilisées dans la norme. Il s'agit en quelque sorte d'une manière d'établir un vocabulaire commun pour les lecteurs et utilisateurs de la norme et une manière de mettre en contexte la norme. Ceci est certainement une idée louable, cependant la liste des définitions doit être rigoureuse, faute de quoi la norme perd de la crédibilité et peut faire en sorte que certains usagers ne pourraient pas l'utiliser, car leur contexte d'utilisation n'est pas compatible avec les définitions fournies par la norme.

La norme ISO/IEC 14764 (ISO, 2006) fournit une définition suivante de la maintenabilité : « La capacité d'un logiciel à être modifié. Les modifications peuvent inclure des corrections, des améliorations ou l'adaptation du logiciel afin de refléter des changements dans l'environnement, dans les besoins ou les spécifications fonctionnelles ».

Bien que cette définition soit vraie, cependant, elle est possiblement incomplète puisqu'elle ne tient pas compte de la notion de qualité dans un contexte de modification ou de maintenance logiciel. Il est facile et rapide de faire une modification, mais la faire correctement selon un objectif de qualité l'est toutefois moins.

Cette notion de qualité est par ailleurs relevée dans l'article « Evaluating the Predelivery Phase of ISO/IEC FDIS 14764 in the Swedish Context» (Kajko-Mattson, Grimlund Glassbrook, & Nordin, Evaluating the Predelivery Phase of ISO/IEC FDIS 14764 in the Swedish Context, 2001). Cet article compare la norme ISO/IEC 14764 avec les pratiques de génie logiciel en Suède. Les auteurs l'article affirme que la norme devrait spécifier un des attributs de qualité en relation avec la notion de maintenabilité.

Bien que nous allons suggérer l'ajout de définir des attributs de qualité lors de la prochaine section de ce chapitre, nous suggérons de débuter par la modification de la définition de la maintenabilité afin d'inclure l'aspect qualité.

Ainsi, la définition de la qualité devrait ressembler à quelque chose comme : « La capacité d'un logiciel à être modifié en fonction d'une qualité attendue. Les modifications peuvent inclure des corrections, des améliorations ou l'adaptation du logiciel afin de refléter des changements dans l'environnement, dans les besoins ou les spécifications fonctionnelles »

L'idée de laisser la notion de qualité relativement libre permet plus de flexibilité aux auteurs de plan de maintenance, car ils pourront définir leurs objectifs de qualité dans leur documentation.

1.2 Qualité

La norme ISO/IEC 9126 définit un modèle de qualité logiciel à l'aide de caractéristiques et de sous-caractéristiques. Bien que ce modèle soit destiné à l'ensemble du domaine du génie logiciel, ces caractéristiques de qualité sont aussi utilisables dans le contexte de maintenance logiciel. Elle définit ainsi la qualité en fonction de la fonctionnalité, de la fiabilité, de l'utilisabilité, de l'efficacité, de la maintenabilité et de la portabilité. (ISO, 2001)

Bien que la norme ISO/IEC 14764 fait souvent référence aux différents attributs de qualité, elle n'est pas incluse dans les définitions situées dans le chapitre 2. Je crois que ceci est oublié significatif à la norme. Je crois que la norme bénéficierait à ajouter une définition de la qualité dans ce chapitre.

Puisque la norme ISO/IEC 9126 existe et représente un incontournable dans le monde du génie logiciel, j'estime qu'il est important que la définition de la qualité soit calquée sur cette dernière.

Je propose donc l'ajout de la définition suivante pour la notion de qualité : « Attributs attendus dans un logiciel développé de manière professionnelle. Ceci est défini dans la norme ISO/IEC 9126 en terme de fonctionnalité, de fiabilité, d'utilisabilité, d'efficacité, de maintenabilité et de portabilité. »

Bien que cette définition à ajouter doit sujet à être débattu, je crois qu'elle comprend quand même trois points importants qui doivent être inclus dans cette définition :

1. Une courte description de la qualité en logiciel
2. Une référence à la norme ISO/IEC 9126
3. La liste des caractéristiques de qualité contenue dans la norme ISO/IEC 9126.

CHAPITRE 2 : MODIFICATIONS DES PROCESSUS

2.1 Mesure

La mesure est une notion importante en génie logiciel. Il existe plusieurs modèles d'estimation afin de mesure la taille fonctionnelle d'une application ou d'une partie d'une application. Bien sûr, ceci est aussi applicable à une modification à être effectuée dans une application.

La littérature abonde en articles et en recherche qui indique et prouve que la mesure a une incidence importante dans la précision des estimées de temps en génie logiciel et, donc, nous permet de mieux évaluer le temps dans le but de pouvoir mieux contrôler le processus de développement logiciel. Le professeur Alain Abran, de l'ÉTS, a écrit plusieurs articles sur le sujet et a même inventé un modèle de mesure, COSMIC-FFP, qui est devenu la norme ISO 19761 (Azzouz & Abran, 2004).

Selon l'article d'Abran et Sellami (Abran & Sellami, Measurement and Metrology Requirements for Empirical Studies in Software Engineering, 2002), la mesure dans la cadre de maintenance logicielle est surtout associée à l'exploitation des résultats de la mesure. Ceci étant dit, la norme ISO/IEC 14764, qui indique les processus qui doivent être effectués afin de faire adéquatement les activités de maintenance logicielle, n'en fait aucune mention. Ceci est d'autant plus vrai dans la section 5.2, qui spécifie les étapes pour l'analyse de la problématique, aucune mention à la mesure n'est effectuée et je crois que ceci est une omission importante. Afin de pouvoir évaluer si une équipe de travail possède les ressources et le temps nécessaire afin de compléter une activité de maintenance logicielle, il est important de pouvoir évaluer la taille de la modification ou de la mise à jour à effectuer.

Ainsi, je suggère de faire une modification au sous-chapitre 5.2.2.1 en ajoutant une nouvelle tâche qui consiste à faire la mesure, en utilisant COSMIC-FFP, de la modification à effectuer.

Cependant, une activité de prise en note, dans une base de données, du temps requis face à la taille fonctionnelle devra aussi être ajouté à la section 5.4.5. Ceci permettra aux analystes de pouvoir faire des analyses comparatives (*« benchmarking »*) afin d'évaluer le temps requis à une modification.

2.2 Méthodes agiles

Pour plusieurs entreprises, les méthodes agiles sont une véritable façon d'être et quand on y pense, ces dernières sont très similaires avec la maintenance logicielle. Les Suédois Svennson et Höst présentent dans leur article les douze principaux concepts des méthodes agiles (Svensson & Höst, 2005). Quand on regarde individuellement chacun des aspects, on peut se rend rapidement compte que plusieurs aspects des activités de maintenance logiciels sont inclus dans les méthodes agiles.

- Planification : Que l'on soit en développement logiciel ou en maintenance, il est important de faire la planification de nos activités.
- Programmation en paire : Dans le contexte où on possède une équipe de maintenance, on devra se partager les tâches afin de partager les connaissances.
- Propriété collective du code source : Dans un contexte d'équipe, c'est une formalité.
- Petite version (« small release ») : C'est typiquement ce qu'on fait en maintenance où on fait plusieurs petites modifications dans le but d'en accomplir une grande.
- Standard de programmation : Dans un contexte d'équipe, c'est une formalité.
- Design simple : Dans le cadre de maintenance, généralement, on tente de garder les choses simples afin de réduire la complexité de l'application et d'en prolonger la vie.
- Métaphore : N'est pas très souvent applicable. Généralement, les équipes travaillent sur plusieurs logiciels dans un environnement complexe.
- Semaine de 40 heures : C'est une pratique difficile à appliquer dans le cadre d'une équipe de maintenance puisque le flot des projets peut être variable.
- Refactoring : Quand on fait plusieurs activités de maintenance, on ne peut omettre ceci puisque ça permet de garder l'application propre et d'en prolonger la vie.

- Tests continus : Un concept à mettre en application si on veut être une organisation qui vise la qualité.
- Client à proximité : Ceci n'est pas toujours possible. Très variable dépendant des organisations.
- Intégration continue : Un concept à mettre en application si on veut être une organisation qui vise la qualité.

Les méthodes agiles correspondent à 9 des 12 concepts. On peut donc affirmer que les méthodes agiles en plus d'être applicable à la maintenance logicielle (Rico, 2008), elles en sont en quelque sorte une sous-catégorie! C'est pour cette raison que je suggère l'ajout d'une section dans le chapitre sur les processus qui discutera des méthodes agiles.

Je suggère donc l'ajout d'un chapitre 5.7 sur les méthodes agiles. Ce chapitre devrait comprendre la liste des aspects que comprends les méthodes agiles et un guide sur comment appliquer chacune d'entre elles. Chaque aspect devrait être le sujet d'un sous-chapitre avec sa propre liste d'entrée et de sorties ainsi que ses différentes tâches à réaliser.

CHAPITRE 3 : AJOUTS DE NOUVELLES CONSIDÉRATIONS D'ÉXÉCUTION

3.1 Environnement de maintenance

Avec chaque nouvelle génération de logiciels vient une nouvelle complexité. Bien sûr ceci se reflète directement sur l'environnement de développement de ces applications qui lui aussi se complexifie. Puisque les équipes de maintenance doivent généralement supporter plusieurs applications sur plusieurs plateformes (Svensson & Höst, 2005), avec le temps ceci peut devenir un réel problème. Pour toute sorte de considération technique, il peut parfois être difficile et long de reproduire l'environnement de développement d'une application traditionnelle (« *legacy application* »).

Ceci est d'autant vrai que les applications modernes sont souvent réparties sur plusieurs serveurs en raison des modèles en couches (Mookerjee, 2005). Il existe plusieurs pistes de solution, tel l'établissement de politique de maintenance conjointe pour l'ensemble des composantes d'un système ou une approche plus architecturale où les systèmes doivent avoir été conçus avec leur maintenance en tête puisqu'ils communiquent par message via un système de bus. Un tel système permet alors de simuler certaines composantes lors de la maintenance (Mookerjee, 2005).

La norme ISO/IEC 14764 n'aborde pas ces sujets.

Je suggère donc l'ajout d'un nouveau sous-chapitre dans la section 6 qui indiquera aux utilisateurs de la norme qu'ils doivent faire un plan de gestion des environnements de développement et de maintenance. Celles-ci doivent être considérées comme des éléments à être traité par la gestion de la configuration.

3.2 Entente de service

Le chapitre 6.3 de la norme ISO/IEC 14764 définit ce qu'est une Entente de service et ce qu'elle devrait inclure. Je juge que cette dernière est incomplète, car plusieurs points sont

incomplets ou pas assez généralisés. À titre comparatif, j'ai trouvé que le chapitre 2.5 du livre « Améliorer la maintenance du logiciel » des auteurs April et Abran était plus complet et fournissait plus de nuances sur les différents types ententes de services possibles.

La norme ISO/IEC 14764 ne traite que d'entente entre un fournisseur de service et un client. Ce type d'entente, bien que très courant, n'est pas le seul type existant. Il existe aussi l'entente de service interne et l'entente d'impartition. (April & Abran, Améliorer la maintenance du logiciel, 2006)

L'entente de service interne représente l'entente entre un groupe d'utilisateur d'une application et le groupe de maintenant de la même organisation qui s'occupe d'effectuer les activités de maintenance sur ladite application.

L'entente d'impartition représente le contrat entre un tiers et une organisation au sujet de la maintenance des logiciels. Généralement, le tiers prend en charge l'ensemble des applications, leur maintenance ainsi que les employés. L'entente se veut aussi d'une durée fixe.

Ces deux types d'entente de maintenance sont très différents du contrat de service avec un tiers. La norme ISO/IEC contient présentement seulement les lignes directrices sur ce type d'entente.

Je suggère donc que le chapitre 6.3 de la norme ISO/IEC soit révisé afin d'inclure une courte description des ententes de services internes et des contrats d'impartition ainsi qu'une liste des différents sujets spécifiques que chaque entente devrait aussi contenir.

De plus, la liste des sujets qui doivent être inclus dans le contrat de service avec un tiers devrait être mise à jour avec d'inclure les sujets suivants :

- Obligations du fournisseur
- Services de la maintenance et services optionnels.

- Obligations du client et de son personnel.
- Reproduction de la documentation et du code
- Limite de responsabilités, force majeure, assignation, survie, lois.

Cette liste provient du livre des professeurs April et Abran. (April & Abran, Améliorer la maintenance du logiciel, 2006)

3.3 Outils

Un autre ajout au chapitre 6 serait l'utilisation potentielle d'outils afin d'aider les utilisateurs de la norme à mieux effectuer leurs activités de maintenance. Le sous-chapitre 6.4 existe déjà, mais est beaucoup trop générique pour être réellement une valeur ajoutée à cette norme.

Bien qu'une norme se veut seulement un guide sur les pratiques à mettre en place, je crois que la mise ne place de certains outils sont en quelque sorte la mise en place de la pratique. Ceci est particulièrement vrai quand on ne peut mettre en place la pratique sans utiliser un outil dû à des considérations techniques. La section 6.4 devrait faire la mention qu'une bonne équipe de maintenance devrait mettre en place les outils et les processus suivants :

- Outil d'intégration continue.
- Outil de tests continus

La littérature abonde d'articles qui font l'éloge de l'intégration continue. Il s'agit d'une méthode relativement simple à mettre en place et permet l'amélioration de la qualité des livrables logiciels (Weber, Helfenberger, & Keller, 2005). De plus, l'ajout de cet outil permet d'améliorer le processus logiciel ; chaque développeur devient responsable de faire l'intégration de son code dans le système. Dans un contexte de maintenance, ceci permet d'éviter une intégration de trop de fonctionnalité à la fin d'une phase et ainsi de corrompre l'application (Holck & Jørgensen, 2003).

Il est une bonne pratique de mettre en place un système de test continu lors de la maintenance, des travaux de recherches en prouvent même l'efficacité (Svensson & Höst,

2005). L'implémentation des outils de test continus permet de faire l'automatisation des tests. Ces dernières viennent à faire partie du processus. Bien que tout ne peut être testé de la même manière, une combinaison de tests unitaires, d'intégration et systémique améliorera la productivité (Weber, Helfenberger, & Keller, 2005).

Je suggère donc l'ajout d'outils d'intégration continue et de tests continus dans le chapitre 6.4 afin d'améliorer la qualité des activés de maintenance.

CHAPITRE 4 : MISE À JOUR DES STRATÉGIES DE MAINTENANCE

4.1 Estimation des coûts

Le chapitre 7.2.4 indique qu'une estimation des coûts de la maintenance à effectuer doit être préparée. Bien sûr que les coûts sont un élément important à considérer lors de la planification d'une maintenance. Une activité avec un coût très dispendieux pour peu de modifications pourrait être retardée afin de faire une autre activité avec un coût moins dispendieux, mais qui permet de faire plus de changement. Tout cela est une question de ratio entre un montant d'argent et un nombre de modifications à effectuer.

Le chapitre 7.2.4 indique ce qui doit être pris en compte lors de l'étape du calcul financier. Que ce soit des éléments comme le voyagement, la formation, les coûts associés aux divers environnements de maintenance et de tests ainsi que les coûts de personnels, tout ceci est pris en compte dans le calcul. Bien que ceci soit fort louable, le véritable coût de la maintenance logiciel n'est pas chiffrable seulement en regardant les montants des dépenses à encourir, mais aussi en regardant la taille des travaux à réaliser. (Boehm, 1984)

Je crois que la vraie méthode pour effectuer le calcul serait d'établir un coût fixe pour chaque modification à établir en points de fonction, ce coût tenant compte des éléments spécifiés dans le chapitre 7.2.4. Et ensuite de faire le calcul du nombre de points de fonctions en utilisant COSMIC-FFP (Azzouz & Abran, 2004), par exemple. Ainsi, le coût d'une activité de maintenance est égal au nombre de points de fonction multiplié par le coût fixé pour chaque point de fonction.

Ceci correspond à ce qui a été écrit par les auteurs Abran et Nguyenkim en 1993 dans un article qui a jeté les premières bases de l'ISO 19761.

Je suggère donc de modifier le chapitre 7.2.4 afin qu'il indique clairement qu'une bonne stratégie d'estimation des coûts serait de trouver le coût de revient de chaque point de

fonction de maintenance. Par la suite, il serait facile de faire l'estimation des coûts en trouvant simplement le nombre de points de fonction que notre activité de maintenance devra contenir.

4.2 Logiciels spécifiques

Le chapitre 7 discute du développement d'une stratégie de maintenance logiciel. Le but de la stratégie est de préparé le terrain, tant au niveau des ressources humaines que matériels, pour la maintenance logiciel.

Le chapitre donne des indications sur le développement de la stratégie sur plusieurs sujets, notamment :

- Le concept de la maintenance
- La portée de la maintenance
- L'estimation du coût
- La planification de la maintenance
- Le plan de maintenance
- Le processus de maintenance

Le problème avec ce chapitre est qu'il ne demande pas de définir de stratégie particulière face à la spécificité de l'application à maintenir. Par exemple, la maintenance dans le cadre d'une application en temps réel, une application SaaS ou d'une application de gestion de comptes étudiants est très différents. Cette différence doit être écrite dans la stratégie et il serait important que la norme ISO/IEC 14764 en fasse mention.

Un peu partout dans la littérature, on peut voir que différents spécialistes en font mention :

- Dans le cadre d'application qui doit fonctionner en temps réel, il est important de faire la planification de la maintenance, il est important d'effectuer les maintenances en respectant la notion de séparation entre la synchronisation du temps *(« timing »)* et les fonctionnalités. (Locke, 1992)

- Dans le cadre d'applications SaaS où des applications de plusieurs clients différents sont sur le même serveur, l'architecture que doit avoir l'application et la qualité de l'implémentation logicielle, le code, sont cruciales (Bezemer & Zaidman, 2010). Dans le cadre de telles applications, ces enjeux devraient être mentionnés dans le plan de test.
- Dans le cadre d'une application web, puisque celle-ci est parfois disponible sur l'internet, la sécurité de l'application est primordiale. Pour ces raisons, les maintenances correctives doivent être une priorité pour toute organisation qui possède ou gère ces applications (Lai, 2011). Le plan de maintenance devra donc en tenir compte et le mentionner.

Il ne s'agit que de quelques exemples parmi plusieurs articles qui en discutent. Par contre, ces exemples démontrent l'importance de la différence entre les applications, un concept dont la norme ne tient pas compte.

Je suggère donc que le chapitre 7.3.2 soit modifié afin d'y inclure que le plan de maintenance doit aussi inclure les conditions particulières de maintenance de l'application. Ces conditions peuvent se rapporter à n'importe quel autre sujet présenté dans le plan de maintenance, mais leurs mentions comme conditions particulières souligneront leur différence et surtout leurs importances dans le cadre de l'application à maintenir et des activités de maintenance à effectuer.

CONCLUSION

En conclusion, je dois avouer que ma perspective face à la norme ISO/IEC a considérablement évolué depuis le début de la rédaction de ce travail. À première vu et lecture, je croyais qu'il s'agissait d'une norme quelque peu dépassée, qui ne reflétait pas les réalités actuelles du monde du génie logiciel. Je pensais même qu'il serait facile, simple et rapide de trouver plusieurs lacunes et correctifs à lui apporter.

Ce ne fût pas le cas et la rédaction de ce travail m'a permis de faire la lecture de plusieurs articles intéressants sur la maintenance logicielle et ça en rapport avec plus perspectives différentes de la maintenance.

La norme ISO/IEC 14764 est une norme bien vivante, bien rédigée et complète. Cependant, quelques modifications pourraient y être intégrées. Ce que je propose représente plutôt des suggestions, voire même des pistes, que les experts mondiaux de la maintenance logicielle devront analyser pendant plusieurs semaines avant de les intégrer dans la norme.

Cependant, je crois que la norme devrait clarifier la définition de la maintenabilité et définir la notion de qualité. De plus, les processus devraient être mis à jour afin de tenir compte de la mesure du logiciel et des méthodes agiles. Les stratégies proposées par la norme devraient faire un meilleur usage de l'estimation des coûts et des spécifiés de divers logiciels. Et finalement, de nouvelles contraintes d'exécution devraient ajouter au sujet des environnements nécessaires à la maintenance, des ententes de services et des outils que requièrent la maintenance.

RECOMMANDATIONS

La liste des recommandations est présentée dans le gabarit de l'ISO/IEC/JTC1.

Le gabarit rempli est disponible dans l'Annexe 1 du présent document.

LISTE DES RÉFÉRENCES

Abran, A., & Nguyenkim, H. (1993). Measurement of the Maintenance Process from a Demand-based Perspective. *Journal of Software Maintenance: Research and Practice* , 63-90.

Abran, A., & Sellami, A. (2002). Measurement and Metrology Requirements for Empirical Studies in Software Engineering. *Measurement and Metrology Requirements for Empirical Studies in Software Engineering.*

April, A., & Abran, A. (2006). *Améliorer la maintenance du logiciel.* Longeuil, Québec, Canada: Loze-Dion.

April, A., Desharnais, J.-M., & Dumke, R. (2005). A formalism of ontology to support a software maintenance knowledge-based system. *Knowledge Creation Diffusion Utilization , 2* (1), 10-16.

Azzouz, S., & Abran, A. (2004). A Proposed Measurement Role in the Rational Unified Process And its Implementation with ISO 19761: COSMIC-FFP. *Software Measurement European Forum - SMEF 2004.* Rome, Italy.

Bezemer, C.-P., & Zaidman, A. (2010). Multi-tenant SaaS applications: maintenance dream or nightmare? *IWPSE-EVOL '10 Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution*, (pp. 88-92). New York, NY, USA.

Boehm, B. W. (1984). Software Engineering Economics. *Software Engineering, IEEE Transactions on , 10* (1), 4-21.

Cozzetti B. de Souza, S., Anquetil, N., & de Oliveira, K. (2005). A Study of the Documentation Essential to Software Maintenance. *SIGDOC '05 Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information.* New York, USA.

Holck, J., & Jørgensen, N. (2003). Continuous integration and quality assurance: A case study of two open source projects. *Australasian Journal of Information Systems, Special Issue , 4*, 40-53.

ISO. (2006). *ISO/IEC 14764:2006 -- Software Engineering -- Software Life Cycle Processes -- Maintenance.*

ISO. (2001). *ISO/IEC 9126-1:2001 -- Software Engineering -- Product quality -- Part 1: Quality model.*

Kajko-Matsson, M., van Deursen, A., Reiger, R., Canfora, G., Ihme, T., Engel, E., et al. (2006). A Model of Maintainability – Suggestion for Future Research. *In Proceedings of Software Engineering Research and Practicev* , 436-441.

Kajko-Mattson, M. (2001). Motivating the Corrective Maintenance Maturity Model (CM3). *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on* , 112-117.

Kajko-Mattson, M., Grimlund Glassbrook, A., & Nordin, M. (2001). Evaluating the Predelivery Phase of ISO/IEC FDIS 14764 in the Swedish Context. *In Proceedings of ICSM* , 431-440.

Koponen, T., & Hotti, V. (2005). Open Source Software Maintenance Process Framework. *ACM SIGSOFT Software Engineering Notes , 30* (4), 1-5.

Lai, S.-T. (2011). Corrective Maintenance based Vulnerability Repair Procedure to Improve Web Application Security. *American Journal of Engineering and Technology Research , 11* (12), 413-416.

Locke, D. C. (1992). Software architecture for hard real-time applications: cyclic executives vs. fixed priority executives. *Real-Time Systems , 4* (1), 37-53.

Mookerjee, R. (2005). Maintaining enterprise software applications. *Communications of the ACM , 48* (11), 75-79.

Rico, D. F. (2008). *http://davidfrico.com.* Récupéré sur http://davidfrico.com/rico08f.pdf

Svensson, H., & Höst, M. (2005). Introducing an Agile Proces in a Software Maintenance and Evolution Organization. *CSMR '05 Proceedings of the Ninth European Conference on Software Maintenance and Reengineering.* Washington, DC, USA.

Weber, R., Helfenberger, T., & Keller, R. K. (2005). Fit for Change: Steps towards Effective Software Maintenance. *Proceedings of the International Conference on Software Maintenance (ICSM'05)*, (pp. 26-33). Budapest, Hungary.

BIBLIOGRAPHY

Abran, A., & Nguyenkim, H. (1993). Measurement of the Maintenance Process from a Demand-based Perspective. *Journal of Software Maintenance: Research and Practice* , 63-90.

Abran, A., & Sellami, A. (2002). Measurement and Metrology Requirements for Empirical Studies in Software Engineering. *Measurement and Metrology Requirements for Empirical Studies in Software Engineering.*

April, A., & Abran, A. (2006). *Améliorer la maintenance du logiciel.* Longeuil, Québec, Canada: Loze-Dion.

April, A., Desharnais, J.-M., & Dumke, R. (2005). A formalism of ontology to support a software maintenance knowledge-based system. *Knowledge Creation Diffusion Utilization , 2* (1), 10-16.

Azzouz, S., & Abran, A. (2004). A Proposed Measurement Role in the Rational Unified Process And its Implementation with ISO 19761: COSMIC-FFP. *Software Measurement European Forum - SMEF 2004.* Rome, Italy.

Bezemer, C.-P., & Zaidman, A. (2010). Multi-tenant SaaS applications: maintenance dream or nightmare? *IWPSE-EVOL '10 Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution*, (pp. 88-92). New York, NY, USA.

Boehm, B. W. (1984). Software Engineering Economics. *Software Engineering, IEEE Transactions on , 10* (1), 4-21.

Cozzetti B. de Souza, S., Anquetil, N., & de Oliveira, K. (2005). A Study of the Documentation Essential to Software Maintenance. *SIGDOC '05 Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information.* New York, USA.

Holck, J., & Jørgensen, N. (2003). Continuous integration and quality assurance: A case study of two open source projects. *Australasian Journal of Information Systems, Special Issue , 4*, 40-53.

ISO. (2006). *ISO/IEC 14764:2006 -- Software Engineering -- Software Life Cycle Processes -- Maintenance.*

ISO. (2001). *ISO/IEC 9126-1:2001 -- Software Engineering -- Product quality -- Part 1: Quality model.*

Kajko-Matsson, M., van Deursen, A., Reiger, R., Canfora, G., Ihme, T., Engel, E., et al. (2006). A Model of Maintainability – Suggestion for Future Research. *In Proceedings of Software Engineering Research and Practicev* , 436-441.

Kajko-Mattson, M. (2001). Motivating the Corrective Maintenance Maturity Model (CM3). *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on* , 112-117.

Kajko-Mattson, M., Grimlund Glassbrook, A., & Nordin, M. (2001). Evaluating the Predelivery Phase of ISO/IEC FDIS 14764 in the Swedish Context. *In Proceedings of ICSM* , 431-440.

Koponen, T., & Hotti, V. (2005). Open Source Software Maintenance Process Framework. *ACM SIGSOFT Software Engineering Notes , 30* (4), 1-5.

Lai, S.-T. (2011). Corrective Maintenance based Vulnerability Repair Procedure to Improve Web Application Security. *American Journal of Engineering and Technology Research , 11* (12), 413-416.

Locke, D. C. (1992). Software architecture for hard real-time applications: cyclic executives vs. fixed priority executives. *Real-Time Systems , 4* (1), 37-53.

Mookerjee, R. (2005). Maintaining enterprise software applications. *Communications of the ACM , 48* (11), 75-79.

Rico, D. F. (2008). *http://davidfrico.com.* Retrieved from http://davidfrico.com/rico08f.pdf

Svensson, H., & Höst, M. (2005). Introducing an Agile Proces in a Software Maintenance and Evolution Organization. *CSMR '05 Proceedings of the Ninth European Conference on Software Maintenance and Reengineering.* Washington, DC, USA.

Weber, R., Helfenberger, T., & Keller, R. K. (2005). Fit for Change: Steps towards Effective Software Maintenance. *Proceedings of the International Conference on Software Maintenance (ICSM'05)*, (pp. 26-33). Budapest, Hungary.

ANNEXE I : GABARIT ISO/IEC/JTC1

No	Page	Texte actuel / Situation	Changement suggéré	Décision
1	3	La définition de la maintenabilité ne tient pas compte du facteur qualité.	Modifier la définition : « La capacité d'un logiciel à être modifié en fonction d'une qualité attendue. Les modifications peuvent inclure des corrections, des améliorations ou l'adaptation du logiciel afin de refléter des changements dans l'environnement, dans les besoins ou les spécifications fonctionnelles.»	
2	4	Ajouter une définition du mot « Qualité »	Ajouter la définition suivante : « Attributs attendus dans un logiciel développé de manière professionnelle. Ceci est défini dans la norme ISO/IEC 9126 en terme de fonctionnalité, de fiabilité, d'utilisabilité, d'efficacité, de maintenabilité et de portabilité. »	
3	11	Les tâches à effectuer dans la maintenance n'incluent pas l'utilisation de la mesure dans les étapes de planifications	Ajouter une étape au chapitre 5.2.2.1 : « Calculer la taille fonctionnelle de la maintenance à effectuer. »	

4	17	L'étape de conclusion d'une activée de maintenance n'inclus pas de prise en note du temps requis en fonction de la taille fonctionnelle.	Ajouter une sortie supplémentaire au chapitre 5.4.5 : « Inscrire le temps qui a été requis à la complétion de l'activité de maintenance ainsi que la taille fonctionnelle dans une base de données ».	
5	24	La norme ne tient pas compte des méthodes agiles, qui sont en quelque sorte une série d'activité de maintenance.	Ajouter un sous-chapitre sur les méthodes agiles où pour chacun des 12 aspects on trouverait un sous-chapitre expliquant ses entrées, sorties et tâches.	
6	26	La norme ne parle que d'un seul type d'entente de service : le contrat de service avec un tiers.	Ajouter au chapitre 6.3 des explications sur deux autres types d'entente de service : l'entente de service et l'entente d'impartition. Pour chacune d'elle, spécifiez la liste des sujets spécifiques qu'elle devra contenir.	
7	26	La liste de ce que doit contenir une entente de service avec un tiers est incomplète.	Ajouter les éléments suivants à ce qu'une entente de service avec un tiers doit contenir : • Obligations du fournisseur • Services de la maintenance et services optionnels. • Obligations du client et de son personnel. • Reproduction de la documentation et du code • Limite de responsabilités, force majeure, assignation, survie, lois.	

8	31	La norme n'aborde pas le thème des environnements de développement et de maintenance.	Ajouter un sous-chapitre qui expliquera aux usagers de la norme qu'ils doivent faire un plan de gestion des environnements de développement et de maintenance. Ceux-ci doivent être traités dans la gestion de la configuration.	
9	26	La norme parle vaguement d'outils de tests et d'outils d'environnement dans le chapitre 6.4.	Je propose que la norme suggère aussi l'utilisation d'outils de tests continus et d'outils d'intégration continue dans le chapitre 6.4 afin d'améliorer la qualité des activités de maintenance.	
10	33	La méthode d'estimation des coûts de la maintenance ne tient pas compte de la taille de la maintenance à effectuer, mais seulement des coûts fixes.	Je suggère d'utiliser les coûts fixes déjà présentés dans la norme pour calculer un coût fixe par rapport à un point de fonction. Une fois ceci fait, on pourra faire l'estimation en comptabilisant les points de fonction que contient une activité de maintenance et en les multipliant par le coût fixe calculé.	

11	33	Le chapitre 7 ne tient pas compte des conditions particulières qu'une application ou un type d'application peut contenir.	Ajouter au chapitre 7.3.2, qui indique ce que doit contenir un plan de maintenance une mention que le plan de mention doit aussi inclure une section qui mets en évidence les spécificités de l'application à maintenir et les impacts de ces différences sur les activités de maintenance.	