LE GUIDE DU CORPUS DES CONNAISSANCES EN GÉNIE LOGICIEL[1]

Pierre Bourque, Robert Dupuis, Alain Abran — Université du Québec à Montréal, James W. Moore — The Mitre Corporation & Leonard Tripp — The Boeing Company

Résumé : Les auteurs, membres de l'équipe directrice du projet SWEBOK, décrivent les trois phases de leur projet, qui vise à caractériser un corpus de connaissances, ce qui est une étape essentielle pour amener le génie logiciel au stade de véritable profession.

L'IEEE Computer Society et l'Association for Computing Machinery collaborent à un projet dont l'objectif est d'élaborer un guide présentant le corpus des connaissances dans le domaine du génie logiciel, le projet SWEBOK (en anglais, Software Engineering Body of Knowledge). La description d'un corpus de connaissances est une étape essentielle vers la reconnaissance d'une profession ; un corpus de ce type permet d'établir un consensus sur le contenu de la discipline concernée. En l'absence d'un tel consensus, on ne peut ni valider un examen qui mène à l'obtention d'une licence, ni instituer un cursus qui prépare les candidats pour cet examen, ni formuler des critères d'accréditation de ce cursus.

Le projet SWEBOK (http://www.swebok.org) en est actuellement à la fin de la deuxième de ses trois phases. Dans cet article, nous en présentons une vue d'ensemble et en résumons les résultats obtenus jusqu'à maintenant.

OBJECTIFS ET PUBLIC VISÉ

Les 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.

Le projet SWEBOK ne cherche pas à établir un corpus de connaissances proprement dit, mais plutôt à mettre sur pied un guide présentant ce corpus. Les connaissances existent déjà ; notre but est de recueillir le consensus sur le noyau de connaissances caractérisant la discipline du génie logiciel.

Pour atteindre ces objectifs, nous avons visé différents publics cibles. D'abord, des organismes publics et privés, qui ont besoin d'une vue cohérente du génie logiciel afin d'en définir l'enseignement et la formation, de catégoriser les emplois et d'établir des politiques d'évaluation de la performance. Ensuite, les praticiens en génie logiciel et les services publics responsables de l'établissement des règles relatives à l'octroi de licences et aux pratiques professionnelles. Enfin, les associations professionnelles et les enseignants qui définissent les règles et les politiques d'accréditation pour les cursus universitaires ainsi que les étudiants qui ont choisi la profession du génie logiciel.

LE GUIDE

Le projet comporte trois phases, appelées Strawman (« homme de paille »), Stoneman (« homme de pierre ») et Ironman (« homme de fer »). Le guide correspondant à la première phase, Strawman, achevé neuf mois après le début du projet, a servi de modèle pour organiser le guide SWEBOK [1]. La version Stoneman sera complétée au printemps 2000. Ironman, la troisième phase, suivra et durera deux ou trois ans. Fondée sur des principes établis dans la deuxième phase, la version Ironman sera enrichie par des analyses plus approfondies, un processus de révision plus large ainsi que par l'expérimentation.

Le guide SWEBOK structurera le corpus de connaissances selon plusieurs domaines de connaissances. Actuellement, la version Stoneman identifie dix domaines. Le tableau 1 présente les spécialistes responsables de la préparation du document descriptif de chacun de ces domaines. Nous envisageons en outre sept disciplines connexes (voir le tableau 2).

Tableau 1

Tableau 2. Les disciplines connexes

Sciences cognitives et facteurs humains
Ingénierie informatique
Informatique
Gestion et sciences de la gestion
Mathématiques
Gestion de projets
Ingénierie de systèmes

Dans le guide, la distinction entre domaines de connaissances et disciplines connexes est importante. Le projet identifiera les domaines de connaissances (ainsi que leurs sujets) qui sont considérés comme fondamentaux pour les ingénieurs en logiciel. Ces ingénieurs devraient connaître des éléments relevant des disciplines connexes, mais le projet SWEBOK ne les spécifiera pas. D'autres s'en chargent, comme le comité de coordination conjoint IEEE Computer Society-ACM pour le génie logiciel, ou le groupe de travail sur l'enseignement du génie logiciel [2].

Comme le montre la figure 1, et comme il est expliqué dans la partie suivante, chaque descriptif de domaine de connaissances (d'une dizaine de pages chacun) contient plusieurs composants importants.

L'organisation hiérarchique

Le guide SWEBOK utilisera une organisation hiérarchique pour décomposer chaque domaine de connaissances. On pourra alors facilement repérer un ensemble de sujets. Une organisation en deux ou trois niveaux permettra aux lecteurs de trouver aisément les sujets qui les intéressent. Le traitement des sujets sera compatible avec celui des principales écoles de pensée et avec celui en usage dans l'industrie, dans la littérature technique et dans les normes relatives au génie logiciel. Le traitement ne présupposera ni domaines d'application, ni utilisations commerciales, ni philosophies de management, ni méthodes de développement en particulier. L'étendue de la description de chaque sujet se limitera à celle qui est nécessaire au lecteur pour trouver les documents de référence. Après tout, le corpus de connaissances se trouve dans les documents de référence et non dans le guide lui-même.

Dès le début du projet, on a soulevé la question du détail du traitement que le guide devait présenter. Après de longues discussions, nous avons adopté le concept de *connaissances généralement admises* [3], connaissances que nous devions distinguer 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 admises s'appliquent à la plupart des projets la plupart du temps et un large consensus permet de valider leur intérêt et leur efficacité.

Cependant, ceci ne signifie pas que nous devrions appliquer les connaissances généralement admises de façon uniforme à tous les travaux relevant du génie logiciel — ce sont les besoins propres de chaque projet qui les déterminent—, mais les ingénieurs en logiciel compétents devraient maîtriser ces connaissances et être en mesure de les appliquer. D'une façon plus précise, ces connaissances sont des éléments qui devraient être étudiés en vue d'un examen conduisant à l'obtention d'une licence en génie logiciel. Les candidats pourront faire cet examen après quatre années de pratique. Spécifique au contexte de l'enseignement aux États-Unis, ce critère ne s'applique toutefois pas nécessairement dans d'autres pays. Mais nous croyons qu'il est tout de même utile et que les deux définitions des connaissances généralement admises devraient être considérées comme complémentaires. Mentionnons enfin que l'organisation proposée est prospective ; ainsi, nous considérons non seulement ce qui est généralement admis aujourd'hui, mais aussi ce qui le sera probablement dans trois ou cinq ans.

Documents de référence et tableau matriciel de croisement

Le guide identifiera des documents de référence pour chacun des domaines de connaissances. Ces documents sont soit des chapitres d'ouvrages, soit des articles ayant été évalués par des pairs ou toute autre source d'information faisant autorité. Les documents de référence doivent être en anglais et facilement accessibles. Nous avons donné la préférence aux documents dont les droits de reproduction sont détenus par l'IEEE Computer Society ou l'ACM, car nous désirons les rendre accessibles gratuitement par Internet.

Le guide comportera aussi une matrice qui reliera les documents de référence à la liste des sujets. Évidemment, une référence donnée pourra s'appliquer à plus d'un sujet.

Classification

Pour permettre d'aborder les sujets de différentes façons et de les relier aux autres disciplines de l'ingénierie, le guide classera les sujets selon la taxinomie des connaissances de la conception en ingénierie

que Walter Vincenti a proposée dans son histoire de l'ingénierie aéronautique publiée en 1990 [4]. Ses six catégories de connaissances relatives à la conception en ingénierie sont respectivement les concepts fondamentaux de la conception, les critères et les spécifications, les outils théoriques, les données quantitatives, les aspects pratiques et les approches de la résolution de problèmes.

Niveaux

Pour aider notamment ceux qui établissent les programmes d'études, le guide attribuera à chaque sujet un niveau correspondant aux catégories pédagogiques proposées par Benjamin Bloom. Selon lui, les objectifs pédagogiques peuvent être classés en six catégories qui représentent chacune un niveau de maîtrise croissant : connaissances, compréhension, application, analyse, synthèse et évaluation. On trouvera la taxinomie de Bloom sur le site Web suivant : http://www.valdosta.peachnet.edu/~whuitt/psy702/cogsys/bloom.html

Les domaines de connaissances des disciplines connexes

Chaque description de domaine de connaissances du guide SWEBOK identifiera aussi les domaines de connaissances des disciplines connexes pertinents. Sans description ni références, l'identification de ces domaines de connaissances devrait malgré tout être utile aux concepteurs de programmes d'études.

LES DOMAINES DE CONNAISSANCES

La sélection, les titres et les descriptions des domaines de connaissances peuvent être revus et corrigés selon les critiques qui auront été formulées. De plus, certains thèmes, comme les mesures, les outils et les normes, sont transversaux et sont actuellement traités de façon séparée dans chaque domaine. Ces décisions seront réexaminées dans les versions ultérieures du guide. Dans cet article, nous présentons, par ordre alphabétique (selon le titre anglais), les domaines de connaissances tels qu'ils sont ébauchés. La figure 2 résume ces dix domaines de connaissances et les sujets importants qui leur sont associés.

Gestion de configuration *(Software Configuration Management)*

Un système peut être défini comme une collection de composants organisés de manière à exécuter une fonction ou un ensemble de fonctions. La configuration d'un système est la fonction ou les caractéristiques physiques du matériel, des microprogrammes, du logiciel, ou une combinaison de ces éléments, énoncés dans la documentation technique et mis en œuvre dans un produit. La gestion de configuration, quant à elle, est la discipline correspondant à l'identification, à certains moments, de la configuration afin d'en contrôler systématiquement les modifications, d'en maintenir l'intégrité et de les retracer tout au long du cycle de vie du système.

Les concepts de la gestion de configuration s'appliquent à tous les éléments requérant un contrôle, bien qu'il y ait des différences entre la mise en œuvre de la gestion de configuration du matériel et celle du logiciel. Les activités principales de la gestion de configuration du logiciel servent de cadre pour l'organisation et la description des sujets de ce domaine de connaissances. Ce sont : la conduite du processus de la gestion de configuration, l'identification, le contrôle, le compte rendu de l'état de la configuration, sa vérification ainsi que la gestion et la livraison des différentes versions du logiciel (voir figure 2a).

Construction du logiciel *(Software Construction)*

La construction du logiciel constitue un acte fondamental du génie logiciel; les programmeurs doivent construire des logiciels qui fonctionnent et qui ont un sens, au moyen de la programmation, de l'autovalidation et des tests unitaires. Loin d'être une simple traduction mécanique d'une bonne conception en un logiciel qui fonctionne, la construction du logiciel concerne étroitement certaines des questions les plus délicates du génie logiciel.

La division de ce domaine de connaissances en sujets est faite selon deux perspectives complémentaires. La première présente trois styles principaux pour les interfaces de la construction du logiciel : linguistique, mathématique et visuel (voir la figure 2b). Puis, pour chaque style, les sujets sont classés selon quatre principes d'organisation de base qui ont une forte influence sur la façon dont la construction du logiciel est effectuée : la réduction de la complexité, l'anticipation de la diversité, la structuration de la validation et l'utilisation de normes externes.

Ainsi, les sujets qui figurent sous la rubrique « anticipation de la diversité », en ce qui concerne les méthodes linguistiques de la

construction du logiciel, sont : le masquage de l'information, la documentation incorporée, les ensembles de méthodes complets et suffisants, l'héritage de classes de l'orienté-objet, la création de langages « colle » pour lier les composants patrimoniaux, le logiciel géré par tables, les fichiers de configuration ainsi que le logiciel et le matériel autodescriptifs.

Conception du logiciel *(Software Design)*

La conception transforme les exigences, habituellement énoncées en des termes appropriés au domaine du problème concerné, en une description qui explique comment résoudre le problème. Elle explique aussi comment le système est décomposé et organisé en composants ainsi que les interfaces entre ces composants. De plus, la conception affine suffisamment la description pour entreprendre leur construction.

Les concepts et les principes de base constituent le premier sous-domaine de ce domaine de connaissances qu'est la conception du logiciel (voir la figure 2c). La qualité de la conception et les mesures en constituent le deuxième sous-domaine, qui comprend les attributs de la qualité, l'assurance-qualité et les mesures. Dans le troisième sous-domaine, l'architecture logicielle, on trouve les structures et les points de vue, les descriptions architecturales, les patrons et les cadres orientés-objet. Ce sous-domaine comprend aussi une partie sur les styles architecturaux — une notion importante en ce qui concerne l'architecture logicielle —, dans laquelle on présente quelques-uns des principaux styles identifiés par différents auteurs.

Le sous-domaine des notations de conception porte sur les notations documentant une conception spécifique de haut niveau ou une conception détaillée de systèmes. Les stratégies et les méthodes de conception, le dernier sous-domaine, touchent quatre sujets principaux, à savoir les stratégies générales, la conception guidée par la structure des données, la conception guidée par les fonctions et la conception orientée-objet.

Infrastructure du génie logiciel *(Software Engineering Infrastructure)*

Ce domaine de connaissances regroupe trois sous-domaines transversaux. Ce sont les méthodes de développement, les outils logiciels et l'intégration de composants (voir la figure 2d).

Les méthodes de développement imposent une structure à l'activité de développement et de maintenance du logiciel dans le but de la rendre systématique et, en fin de compte, plus . Ces méthodes fournissent habituellement une notation et un lexique, des procédures qui permettent d'effectuer des tâches identifiables ainsi que des lignes directrices permettant de vérifier à la fois le processus et le produit. La portée des méthodes de développement varie largement, allant d'une phase du cycle de vie isolée au cycle complet. Le guide SWEBOK divisera ce sous-domaine en trois sujets principaux liés entre eux : les méthodes heuristiques se rapportant aux approches informelles, les méthodes formelles concernant les approches mathématiques et, enfin, les méthodes de prototypage concernant les approches qui reposent sur diverses formes de prototypage.

Les outils logiciels sont informatisés et destinés à étayer le processus de l'ingénierie du logiciel. Ces outils sont souvent conçus pour supporter certaines méthodes et réduire la charge administrative qui accompagne habituellement leur mise en œuvre manuelle. Comme les méthodes, ils ont pour but de rendre le développement plus systématique, et leur portée va du support à des tâches individuelles à la prise en compte du cycle de vie complet. Le plus haut niveau de la décomposition du domaine des outils logiciels permet de faire la distinction entre les outils conçus pour le développement et la maintenance, ceux pour les activités de support et, enfin, pour le management. Les autres catégories recouvrent les ensembles d'outils intégrés (nommés aussi « environnements de génie logiciel ») et les techniques d'évaluation d'outils.

L'émergence de composants logiciels comme approche viable du développement du logiciel traduit une maturation de la discipline qui vainc le syndrome du non-inventé soi-même. Le sous-domaine de l'intégration des composants est divisé en sujets concernant les composants individuels, les modèles de référence qui décrivent comment les composants peuvent être combinés et aborde le sujet plus général de la réutilisation.

Management du génie logiciel *(Software Engineering Management)*

Le domaine de connaissances du management de l'ingénierie du logiciel regroupe le sous-domaine du processus du management et celui des mesures (voir la figure 2e). Ces deux sous-domaines sont souvent considérés comme distincts (et sont

en général enseignés comme tels). Mais ceux-ci, tout en étant uniques, ont des relations étroites, ce qui a motivé leur traitement combiné. Car, essentiellement, un management sans mesures, qualitatives ou quantitatives, indique un manque de rigueur, et des mesures sans management indiquent, elles, une absence d'objectifs ou de contexte.

Le sous-domaine du processus de management considère la notion de management « à grande échelle » dans le thème de la coordination, qui se rapporte à des questions comme le choix de projets, le développement et l'implantation de normes ainsi que la constitution d'équipes et le développement en équipe. Dans ce sous-domaine sont présentés aussi les sujets restants en fonction des étapes dans le cycle de vie du développement du projet : démarrage et définition de la portée du projet, planification (y compris le calendrier et l'évaluation des coûts et des risques), exécution et, finalement, revue, évaluation et clôture du projet.

Le sous-domaine des mesures regroupe quatre sujets : les buts du programme de mesure, le choix des mesures, la cueillette des données et le développement des modèles. Les trois premiers concernent principalement la théorie et l'utilité des mesures et traitent, entre autres, du choix de celles-ci. Il est aussi question de la prise de mesures, qui comporte à la fois des questions techniques (comme l'extraction automatique) et des questions humaines (la conception des questionnaires et la réponse aux mesures effectuées). Le quatrième sujet (le développement de modèles) traite de l'utilisation des données et des connaissances nécessaires pour construire des modèles.

Processus du génie logiciel *(Software Engineering Process)*

Ce domaine de connaissances recouvre la définition, la mise en œuvre, la mesure, le management, la modification et l'amélioration des processus logiciels. Dans le premier sous-domaine, les concepts de base et les définitions, sont établis les thèmes et la terminologie (voir la figure 2f).

Le but et les méthodes pour définir des processus logiciels, comme les différentes définitions des processus et leur support automatisé sont décrits dans le sous-domaine de la définition des processus. Différents thèmes y figurent : les types de définitions de processus, les modèles de cycle de vie, les modèles de processus de cycle de vie, les notations pour les définitions des processus, les méthodes de définition de processus et l'automatisation.

Le sous-domaine de l'évaluation des processus décrit les approches des analyses quantitative et qualitative. La mesure étant importante dans l'évaluation des processus, la méthodologie pour la mesure du processus est le premier sujet abordé dans ce sous-domaine. Un paradigme analytique et un paradigme d'évaluation des performances divisent les types d'évaluation. Le paradigme analytique repose sur une preuve quantitative pour déterminer où une amélioration est nécessaire ou si une amélioration a porté ses fruits. Sous ce paradigme, on trouve l'évaluation qualitative, l'analyse des causes premières, la simulation des processus, la classification orthogonale des défauts, les études expérimentales et le processus logiciel individuel. Le paradigme d'évaluation des performances repose, quant à lui, sur l'identification d'une organisation qui fait preuve d'excellence dans un domaine afin de documenter ses pratiques et outils. Les modèles et méthodes d'évaluation de processus sont les deux sujets principaux de ce paradigme.

Le sous-domaine de la mise en œuvre et de la modification de processus décrit les paradigmes, l'infrastructure et les facteurs nécessaires à cette mise en œuvre et à la réussite de la modification de processus. On trouve, dans ce sous-domaine, les paradigmes pour la mise en œuvre et la modification des processus, l'infrastructure, les lignes directrices ainsi que l'évaluation de la mise en œuvre et de la modification de processus.

Évolution et maintenance du logiciel *(Software Evolution and Maintenance)*

La maintenance du logiciel est définie par la norme IEEE 1219-1998 (IEEE Standard for Software Maintenance) comme étant la modification apportée à un produit logiciel après sa livraison pour en corriger des défauts ou améliorer ses performances ou d'autres attributs, ou encore adapter le produit à un environnement modifié. Cependant, les systèmes logiciels sont rarement figés et évoluent constamment dans le temps. Par conséquent, seront abordés dans ce domaine de connaissances les sujets touchant l'évolution du logiciel.

Le sous-domaine des concepts de la maintenance définit la maintenance et décrit ses concepts de base ainsi que la place qu'occupe le concept d'évolution de système dans le génie logiciel (voir la figure 2g). Il explique aussi les tâches que les équipes de

maintenance ont à assumer. Le sous-domaine des rôles et des activités de maintenance porte sur les types formels de maintenance et les activités courantes. Comme pour le développement du logiciel, le processus est crucial pour la réussite et la compréhension de l'évolution du logiciel et de sa maintenance. Le sous-domaine suivant porte sur les processus de maintenance standard. L'organisation de la maintenance peut en effet être différent de celle du développement ; le sous-domaine relatif aux aspects organisationnels traite précisément de ces différences.

Comme on peut le voir dans le sous-domaine relatif aux problèmes de la maintenance, l'évolution et la maintenance du logiciel présentent pour le génie logiciel différents problèmes techniques et managériaux. Le coût est toujours capital lorsqu'il est question d'évolution et de maintenance du logiciel. Le sous-domaine relatif aux coûts de la maintenance et à l'estimation de ces coûts concerne ceux du cycle de vie du logiciel autant que ceux des tâches individuelles qui se rapportent à ces deux fonctions. Le sous-domaine concernant la mesure relative à la maintenance traite des mesures et de la qualité. Le dernier sous-domaine, celui des outils et des techniques de la maintenance, rassemble de nombreux sous-thèmes que la description du domaine de connaissances n'aborderait pas autrement.

Analyse de la qualité du logiciel *(Software Quality Analysis*

L'élaboration de produits de qualité est la clé de la satisfaction du client. Des logiciels sans les caractéristiques et le niveau de qualité requis indiquent un échec ou une insuffisance du génie logiciel. Cependant, même dans le cas des meilleurs processus logiciels, la spécification des exigences peut ne pas répondre aux besoins du client, le code, ne pas correspondre aux exigences, et des erreurs peuvent rester subtilement cachées jusqu'à ce qu'elles causent des problèmes mineurs ou majeurs, voire des pannes catastrophiques. Ce domaine de connaissances traite de ce qui est relatif à l'assurance-qualité du logiciel et aux activités de vérification et de validation qui y sont liées.

Le but du génie logiciel est de faire un produit de qualité, mais la qualité en soi peut avoir plusieurs significations. Malgré des terminologies différentes, il y a, pour tout un éventail de produits, un certain consensus sur les définitions de la qualité et la sécurité de fonctionnement du logiciel. Ces définitions offrent les connaissances de base à partir desquelles des produits de qualité sont planifiés, construits, analysés, mesurés et améliorés. Le sous-domaine de la définition des produits de qualité examine ces définitions (voir la figure 2h).

L'assurance-qualité est un processus conçu pour assurer la qualité du produit; il s'agit d'un cadre planifié et systématique où sont définies toutes les actions nécessaires pour s'assurer de la conformité du produit à ses exigences techniques. La vérification et la validation du logiciel mènent à une évaluation objective des produits et des processus logiciels tout le long du cycle de vie du logiciel ; en d'autres termes, le processus de vérification et de validation permet au management de contrôler la qualité du produit.

Ces deux processus constituent les fondements du sous-domaine de l'analyse de la qualité du logiciel, sous-domaine qui se subdivise en quatre : la définition de l'analyse de la qualité, les plans des processus, les activités et les techniques de l'analyse de la qualité et, enfin, les mesures dans l'analyse de la qualité du logiciel.

Analyse des exigences du logiciel *(Software Requirements Analysis)*

Le domaine de connaissances de l'analyse des exigences du logiciel est divisé en cinq sous-domaines qui correspondent approximativement aux tâches du processus, qui sont souvent exécutées en parallèle et itérativement plutôt que séquentiellement (voir la figure 2i).

Le sous-domaine du processus d'ingénierie des exigences introduit le processus de l'ingénierie des exigences, oriente les quatre sous-domaines suivants et montre comment l'ingénierie des exigences s'insère dans le cycle de vie global du logiciel. Ce chapitre traite aussi des questions contractuelles et organisationnelles relatives aux projets.

Le sous-domaine de la clarification des exigences recouvre ce qu'on appelle aussi saisie, découverte ou acquisition des exigences. Dans ce sous-domaine, il est question de l'origine des exigences et de la façon dont elles peuvent être colligées par les ingénieurs spécialistes du logiciel. Il s'agit de la première étape dans l'établissement de la compréhension du problème que le logiciel doit résoudre. C'est une activité fondamentalement humaine dont le but est d'identifier les dépositaires des enjeux et d'établir les relations entre l'équipe de développement et le client.

Le sous-domaine de l'analyse des exigences concerne le processus consistant à vérifier les exigences pour y détecter les conflits possibles et les résoudre, découvrir les limites du système et la façon dont celui-ci doit interagir avec son environnement et passer des exigences du client à celles du logiciel.

Le sous-domaine de la validation des exigences vérifie que celles-ci ne comportent ni omissions, ni conflits, ni ambiguïtés et s'assure qu'elles suivent les normes de qualité prescrites. Les exigences devraient être obligatoires, suffisantes et être décrites de façon à minimiser les fausses interprétations.

Le sous-domaine du management des exigences recouvre le cycle de vie du logiciel dans son entier. Il s'agit fondamentalement d'en gérer les modifications et de maintenir les exigences de façon à refléter le logiciel qui a été ou sera construit.

Test du logiciel *(Software Testing)*

Le test du logiciel consiste à vérifier dynamiquement, par le biais d'un ensemble fini de jeux d'essai, le comportement réel d'un programme par rapport au comportement spécifié et attendu. Les jeux d'essai sont sélectionnés, de manière adéquate, à partir du

domaine habituellement infini d'exécutions possibles du programme. Ce procédé d'évaluation ainsi que d'autres concepts de base et définitions constituent le premier sous-domaine (voir la figure 2j).

Ce domaine de connaissances divise le sous-domaine des niveaux de test en deux parties orthogonales, la première étant organisée selon les phases traditionnelles de test des grands systèmes logiciels ; la deuxième est consacrée aux tests pour des conditions ou des propriétés particulières.

Le sous-domaine suivant décrit les connaissances relatives à plusieurs techniques de test généralement admises. Ces techniques sont classées à partir de leurs fondements : elles sont basées sur les spécifications, sur le code, sur les défauts, sur l'utilisation ou sont spécialisées. Ce domaine de connaissances traite des mesures liées aux tests dans leur propre sous-domaine. Le suivant aborde les questions relatives à l'organisation et au contrôle du processus de test, y compris les problèmes de management et les activités de test. Le sous-domaine du test automatique porte sur les outils existants et les concepts liés à l'automatisation du processus de test.

LE PROJET

Depuis 1993, l'IEEE Computer Society et l'ACM collaborent à la promotion de la professionnalisation du génie logiciel par leur participation au comité de coordination sur le génie logiciel — Software Engineering Coordinating Committee ou SWECC (site Web : http://computer.org/tab/swecc).

La portée du projet SWEBOK, la diversité des communautés qui y sont engagées et la nécessité d'une participation importante requièrent un management à temps plein plutôt que reposant sur le volontariat. C'est pourquoi le SWECC a donné le mandat de la gestion du projet au Laboratoire de recherche en gestion des logiciels (LRGL) de l'Université du Québec à Montréal. Le LRGL est supervisé par le SWECC.

L'équipe du projet a énoncé deux principes importants : transparence et consensus. Par souci de transparence, en effet, le processus de développement est lui-même documenté, publié et diffusé pour faire en sorte que les décisions importantes et l'avancement des travaux soient connus par toutes les parties concernées. En outre, nous croyons qu'un consensus est la seule méthode pratique pour légitimer notre guide ; ainsi, il doit y avoir une forte participation et un accord de tous les secteurs de la communauté. Au moment où la version Stoneman du guide sera terminée, des centaines de collaborateurs et de relecteurs auront participé au projet d'une façon ou d'une autre.

Les collaborateurs du projet

À l'instar de tout projet logiciel, le projet SWEBOK présente plusieurs dépositaires d'enjeux — certains d'entre eux étant officiellement représentés. Un comité consultatif de l'industrie (Industrial Advisory Board ou IAB), composé de représentants de l'industrie (Boeing, le National Institute of Standards and Technology, le Conseil national de recherches du Canada, Raytheon et SAP Labs Canada) et d'associations professionnelles (IEEE Computer Society et ACM), apporte au projet son aide financière. Grâce à la générosité de l'IAB, les produits du projet sont publics et gratuits. Ils peuvent être consultés sur le site Web : http://www.swebok.org. À la participation de l'IAB s'ajoutent celles d'organismes de normalisation (IEEE Software Engineering Standards Committee et ISO/IECJTC1/SC7) et de projets connexes (Computing Curricula 2001 Initiative). L'IAB vérifie et approuve les plans du projet, veille à la bonne marche des processus d'établissement du consensus et de relecture fait la promotion du projet et lui donne de la crédibilité. De façon générale, il garantit l'adéquation des travaux aux besoins du monde réel.

Toutefois, nous sommes conscients qu'un corpus de connaissances implicite existe déjà dans les ouvrages d'enseignement sur le génie logiciel. Aussi, pour s'assurer que nous caractérisons correctement la discipline, Steve McConnell, Roger Pressman et Ian Sommerville, les auteurs de trois ouvrages à succès sur le génie logiciel, ont accepté de faire partie d'un comité d'experts représentant la voix de l'expérience. De plus, le processus de relecture (décrit plus loin) permet aux communautés concernées de s'exprimer sur le projet. Rappelons ici que nous visons une participation internationale.

La littérature normative

Le rapport du projet à la littérature normative diffère des travaux antérieurs. La majeure partie de la littérature relative au génie logiciel fournit des informations utiles aux ingénieurs spécialistes du logiciel, mais une faible proportion en est normative. Un document normatif prescrit ce qu'un ingénieur doit faire plutôt que la diversité des choses que cet ingénieur peut ou pourrait faire. La littérature normative est validée par

le consensus établi entre les praticiens et est concentrée sur les normes et documents connexes.

Dès le début, le projet SWEBOK a été conçu pour qu'il soit en relation étroite avec la littérature normative du génie logiciel. Les deux organismes principaux de normalisation pour le génie logiciel y sont ainsi représentés. En fait, une première ébauche des domaines de connaissances reposait directement sur les 17 processus décrits dans la norme ISO/IEC 12207 (Technologie de l'information - Processus du cycle de vie des logiciels). Nous espérons que, ultimement, les normes relatives à la pratique du génie logiciel et leurs principes se retrouvent aisément dans le guide SWEBOK.

Relectures

Nous avons organisé l'élaboration de la version Stoneman en trois cycles de relecture. Le premier cycle s'est concentré sur la pertinence de la division par sujets proposée pour chaque domaine de connaissances. Trente-quatre experts ont participé à cette relecture en avril 1999. Les commentaires des relecteurs ainsi que leur identité sont disponibles sur le site Web du projet.

Le deuxième cycle a été organisé autour des consignes que nous avions données à l'origine aux spécialistes des domaines de connaissances. Un groupe beaucoup plus important de professionnels, mis sur pied à partir de différents points de vue de relecture, a répondu à un questionnaire détaillé pour chaque description de domaine de connaissances. Les points de vue (par exemple, ceux des praticiens, des enseignants et des responsables de politiques publiques) ont été formulés de façon à s'assurer de l'adéquation du guide aux divers publics visés. Les commentaires des relecteurs de ce cycle, qui a pris fin en octobre 1999, sont aussi disponibles sur le site Web du projet. Les spécialistes des domaines de connaissances décriront à leur tour comment les commentaires des relecteurs ont été pris en compte dans les descriptions des domaines de connaissances.

Le troisième cycle de relecture portera sur la justesse et l'utilité du guide. Ce cycle, dont le démarrage est prévu pour janvier 2000, sera effectué par des personnes et des organismes définissant un profil des groupes d'intérêt potentiels (voir l'encadré « Comment contribuer au projet »). Nous avons déjà recruté des centaines de professionnels pour relire le guide au complet et nous en sollicitons davantage pour en assurer la couverture complète.

Conclusion

Tout au cours du projet, l'équipe SWEBOK a rendu disponibles des documents décrivant concrètement l'avancement du projet. La plupart de ces documents sont publics et peuvent être consultés sur le site Web du projet. (Par courtoisie envers les spécialistes des domaines de connaissances, les documents provisoires ne seront pas diffusés avant leur achèvement.) L'équipe du projet met actuellement à jour les descriptions des domaines de connaissances en tenant compte des résultats du deuxième cycle de relecture. Au tout début de l'an 2000, nous inviterons les associations professionnelles et la communauté du génie logiciel à participer au troisième cycle de relecture et à faire part de leurs commentaires sur l'ensemble du guide. La version Stoneman du guide ainsi terminée sera disponible sur le site Web au cours du printemps 2000 et, dans la mesure du possible, les documents de référence cités seront gratuits.

Avant de développer la version Ironman du guide, nous utiliserons la version Stoneman dans le cadre d'applications expérimentales afin d'en déterminer la facilité d'utilisation. Bien que la couverture visée soit identique, le développement d'Ironman aura recours à un processus de relecture plus large et plus exhaustif ; ce processus reposera en effet sur des utilisations expérimentales du guide.

REMERCIEMENTS

L'équipe du projet SWEBOK tient à remercier les membres du comité industriel consultatif pour leur support. Le financement du projet est assuré par l'Association for Computing Machinery, Boeing, Conseil national de recherches du Canada, l'IEEE Computer Society, le National Institute of Standards and Technology, Raytheon et SAP Labs (Canada). L'équipe tient à remercier aussi les spécialistes des domaines de connaissances pour leur travail considérable. Enfin, l'équipe est reconnaissante aux relecteurs pour leur indispensable contribution.

Les auteurs tiennent également à remercier M. Jean-Claude Rault, Mme Sybille Wolff et Mme Louise Tremblay qui ont tous contribué à la traduction de cet article.

Les lecteurs peuvent contacter les auteurs en s'adressant à James W. Moore à l'adresse suivante :
The MITRE Corp., 1820 Dolley Madison Blvd., W534, McLean, VA 22102, États-Unis ;

Courriel : james.w.moore@ieee.org

RÉFÉRENCES

[1] P. Bourque et coll., *Guide to the Software Engineering Body of Knowledge: A Strawman Version*, Université du Québec à Montréal, 1998, http://www.swebok.org .

[2] D.J. Bagert et coll., *Guidelines for Software Engineering Education, Version 1.0*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, novembre 1999 http://www.sei.cmu.edu/collaborating/ed/workgroup-ed.html .

[3] Project Management Institute, *A Guide to the Project Management Body of Knowledge*, Upper Darby, Pennsylvanie, 1996 ; http://pmi.org/publictn/pmboktoc.htm

[4] W. Vincenti, *What Engineers Know and How They Know It: Analytical Studies from Aeronautical History*, The Johns Hopkins University Press, Baltimore, 1990.

COMMENT CONTRIBUER AU PROJET

Les lecteurs souhaitant participer au troisième cycle de relecture qui portera sur l'ensemble du guide du corpus des connaissances sur le génie logiciel peuvent s'inscrire sur le site Web du projet (http://www.swebok.org). Le passage de la version Stoneman à la version Ironman reposera principalement sur les réactions aux applications expérimentales du guide Stoneman. Les personnes intéressées par ces applications sont invitées à contacter Pierre Bourque (bourque.pierre@uqam.ca).

Tableau 1. Les domaines de connaissances du projet SWEBOK et leurs spécialistes

Domaine de connaissances	Spécialistes
Gestion de configuration du logiciel	John A. Scott et David Nisse, Lawrence Livermore Laboratory, États-Unis
Construction du logiciel	Terry Bollinger, The Mitre Corporation, États-Unis
Conception du logiciel	Guy Tremblay, Université du Québec à Montréal, Canada
Infrastructure du génie logiciel	David Carrington, University of Queensland, Australie
Management du génie logiciel	Stephen G. MacDonnell et Andrew R. Gray, University of Otago, Nouvelle-Zélande
Processus du génie logiciel	Khaled El Eman, Conseil national de recherches , Canada
Évolution et maintenance du logiciel	Thomas M. Pigoski, Techsoft, États-Unis
Analyse de la qualité du logiciel	Dolores Wallace et Larry Reeker, National Institute of Standards and Technology, États-Unis
Analyse des exigences du logiciel	Pete Sawyer et Gerald Kotonya, Lancaster University, Grande-Bretagne
Test du logiciel	Antonia Bertolino, Consiglio Nazionale delle Ricerche, Italie

Figure 1. L'organisation de la description d'un domaine de connaissances

Division hiérarchique des sujets
Tableau matriciel des sujets et des documents de référence
Documents de référence
Descriptions des sujets
Classification selon la taxinomie de Vincenti
Catégories pédagogiques selon la taxinomie de Bloom
Références relatives aux disciplines connexes

Figure 2. Une représentation visuelle du guide

LE GUIDE DU CORPUS DES CONNAISSANCES EN GÉNIE LOGICIEL

GESTION DE CONFIGURATION DU LOGICIEL

- Conduite du processus de gestion de configuration
- Identification de la configuration logicielle
- Contrôle de la configuration logicielle
- Compte rendu de l'état de la configuration logicielle
- Vérification de la configuration logicielle
- Gestion et livraison des versions du logiciel

(a)

CONSTRUCTION DU LOGICIEL

- Méthodes de construction linguistiques
- Méthodes de construction mathématiques
- Méthodes de construction visuelles

- Réduction de la complexité
- Anticipation de la diversité
- Structuration pour la validation
- Utilisation de normes externes

(b)

CONCEPTION DU LOGICIEL

- Concepts de base et principes
- Qualité de la conception et mesures
- Architecture logicielle
- Notations de conception
- Stratégies et méthodes de conception

(c)

INFRASTRUCTURE DU GÉNIE LOGICIEL

- Méthodes de développement
Méthodes heuristiques
Méthodes formelles
Méthodes de prototypage
- **Outils logiciels**
Outils de développement et de maintenance
Outils pour activités de support
Outils de management
Ateliers : outils intégrés et environnements de génie logiciel
Techniques d'évaluation des outils
- Intégration des composants
Définition des composants
Modèles de référence
Réutilisation

(d)

MANAGEMENT DU GÉNIE LOGICIEL

- Processus du management
Coordination
Démarrage et définition de la portée
Planification
Exécution
Revue et évaluation
Clôture
- Mesure

(e)

PROCESSUS DU GÉNIE LOGICIEL

- Concepts de base et définitions
Thèmes
Terminologie
- Définition des processus
Définitions des types de processus
Modèles de cycle de vie
Modèles de processus de cycle de vie
Notations pour les définitions de processus
Méthodes de définition des processus

Automatisation
- Évaluation des processus
Méthodologie de la mesure des processus
Paradigme analytique
Paradigme de l'évaluation des performances
- Mise en œuvre et modification des processus
Paradigmes pour la mise en œuvre et la modification des processus
Infrastructure
Consignes pour la mise en œuvre et la modification des processus
Évaluation de la mise en œuvre et de la modification des processus

(f)

ÉVOLUTION ET MAINTENANCE DU LOGICIEL

- Concepts de la maintenance
Activités et rôles de la maintenance Processus de la maintenance Aspect organisationnel de la maintenance
Problèmes de la maintenance du logiciel
Coût de la maintenance et estimation du coût de la maintenance
Mesures de la maintenance
- Outils et techniques pour la maintenance

(g)

ANALYSE DE LA QUALITÉ DU LOGICIEL

- Définition de la qualité des produits
Caractéristiques de la qualité selon la norme ISO 9126
Sécurité de fonctionnement
Volets de la qualité relatifs aux processus et aux contextes particuliers
- Analyse de la qualité du logiciel
Définition de l'analyse de la qualité
Plans des processus
Activités et techniques de l'analyse de la qualité
Mesure en matière d'analyse de la qualité du logiciel

(h)

ANALYSE DES EXIGENCES DU LOGICIEL

- Processus de l'ingénierie des exigences
- Clarification des exigences
- Analyse des exigences
- Validation des exigences
- Management des exigences

(i)

TEST DU LOGICIEL

- Concepts de base et définitions
- Niveaux de test
- Techniques de test
- Mesures relatives au test
- Organisation et contrôle du processus de test
- Automatisation du test

(j)