UNIVERSITÉ DU QUÉBEC À MONTRÉAL

LA PLACE DE LA MESURE
AU SEIN DES PRINCIPES FONDAMENTAUX
DU GÉNIE LOGICIEL

PROPOSITION D'ACTIVITÉ DE SYNTHÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE DE GESTION

PAR
SIBYLLE WOLFF

OCTOBRE 1998

TABLE DES MATIÈRES

LISTE DES FIGURES ET DES TABLEAUX

INTRODUCTION

Depuis son apparition en 6000 avant Jésus-Christ, la science du génie n'a cessé de contribuer à l'amélioration de la qualité de vie des hommes, que ce soit au niveau de la santé, de la richesse, des conditions de travail, etc. Il n'est pas présomptueux d'affirmer que le génie constitue une partie essentielle de l'évolution de l'humanité. Aujourd'hui encore, nous constatons aisément l'effet du génie sur notre vie. Cependant, à l'aube de l'an 2000, d'autres domaines tels que l'informatique et les télécommunications influencent également notre quotidien. Au moyen de l'Internet et du courrier électronique, ces deux disciplines diminuent considérablement les distances. Elles facilitent également la globalisation des marchés. L'informatique en particulier occupe de plus en plus une place indispensable dans les entreprises en permettant de traiter les données de plus en plus rapidement et en offrant à certaines entreprises des avantages concurrentiels. Par contre, le nombre croissant de logiciels crée des problèmes. Ces logiciels sont souvent mal conçus, inadéquats et non conviviaux. Dans le but de résoudre ces problèmes, une nouvelle discipline a été créée, celle du génie logiciel. Bien que le nom laisse supposer que le génie logiciel constitue une branche du génie, cette discipline n'a pas encore été acceptée officiellement par les ingénieurs. De par son jeune âge, cette discipline n'est pas encore bien définie.

Lors du deuxième «International Symposium on Software Engineering Standards» (ISESS'95), une personnalité du génie logiciel, M. David Lorge Parnas, a suggéré de tenter de déterminer les principes fondamentaux du génie logiciel. Nous devons à M. Parnas la création d'un grand nombre de concepts du génie logiciel. Son discours à l'ISESS'95 a eu des répercussions importantes et a donné naissance à des projets de recherche sur les principes fondamentaux du génie logiciel.

L'identification de ces principes, comme nous le verrons dans le reste du document, permettrait ainsi de s'attaquer à des problèmes plus précis, dont, entre autres, l'établissement de normes cohérentes et fondées et la définition précise de la discipline du génie logiciel comme branche du génie. Lors du «Forum on Software Engineering Standards Issues» de 1996 (SES'96), deux participants de l'Université du Québec à Montréal ont décidé de s'attarder à la tâche d'identification des principes fondamentaux du génie logiciel. Depuis, ils ont effectué plusieurs démarches en ce sens. Ces démarches sont décrites brièvement dans les paragraphes suivants.

Dans un atelier de travail durant la conférence SES de 1996, les deux responsables du projet, ainsi que les autres participants, ont tenté de définir des critères pour l'établissement de principes fondamentaux en génie logiciel. Ils sont arrivés à un ensemble de huit critères auxquels devaient répondre les principes (Jabir, 1998).

Au printemps 1997, ces deux chercheurs ont réalisé une étude Delphi auprès de quatorze personnalités très connues du génie logiciel. Cette étude a permis d'établir une liste de principes.

Par la suite, un deuxième atelier s'est tenu durant la conférence ISESS de juin 1997. Durant cet atelier, des propositions de modification, de suppression et d'ajout de critères et de principes ont été soumises.

Lors du deuxième atelier, il a été suggéré d'effectuer une étude semblable auprès de membres de la IEEE Computer Society et plus précisément auprès de membres impliqués dans le génie logiciel. Une partie de notre recherche consiste à réaliser cette deuxième étude Delphi. Celle-ci devrait permettre d'épurer les résultats obtenus lors de la première étude. Puis, nous tenterons de souligner les corrélations entre notre étude et un sondage réalisé auprès de praticiens en génie logiciel de la IEEE Computer Society.

Bien que ces recherches soient effectuées dans le but de faire avancer la discipline du génie logiciel, nous allons plutôt nous concentrer sur l'aspect mesure en génie logiciel Nous voulons identifier les principes fondamentaux du génie logiciel, afin de pouvoir y évaluer la place qu'accordent les experts à la mesure. Ainsi, au lieu de considérer cette étude d'un point de vue uniquement général, nous orienterons notre recherche principalement vers l'aspect de la mesure en génie logiciel. En évaluant ainsi le rôle de la mesure au sein des principes fondamentaux du génie logiciel, nous comptons faire une première tentative d'évaluation du degré de maturité de cette discipline.

Dans les prochaines sections, nous vous décrirons la recherche que nous comptons effectuer. Une description de nos objectifs suivra l'exposé de la problématique adoptée dans cette recherche. Par la suite, nous préciserons les différentes méthodes utilisées dans notre travail. Ces méthodes couvrent le cadre conceptuel, les méthodes de cueillette de données et les méthodes de traitement de données ainsi que les démarches nous permettant d'atteindre nos objectifs. Nous terminerons ce document par un échéancier et une brève conclusion.

1. PROBLÉMATIQUE

L'expression «génie logiciel» est née en 1968 d'un atelier de l'OTAN, lequel devait traiter de l'état et des perspectives de la production de logiciels (Shaw, 1990). Depuis, cette expression est devenue très à la mode. Elle désigne actuellement l'application d'une approche systématique, disciplinée et quantifiable au développement, à l'opération et à la maintenance du logiciel, c'est-à-dire l'application du génie au logiciel (IEEE 610.12, 1990). Cependant, comme nous l'avons mentionné plus haut, on est en droit de se demander si le génie logiciel est réellement une branche du génie, ou s'il n'y a pas un abus du terme. Cela nous amène à parler de plusieurs problèmes auxquels sont confrontées les personnes qui oeuvrent dans ce domaine.

L'aspect que nous abordons en premier lieu et qui va nous intéresser principalement au cours de notre recherche, est celui de la mesure. L'étude de cet aspect pourrait permettre de faire avancer les démarches concernant une série de problèmes auxquels le génie logiciel est confronté. Ces problèmes sont dus au fait que le génie logiciel est encore si jeune et donc mal défini. En effet, il est difficile de définir ce qu'est un ingénieur en logiciel et quels sont ses devoirs envers ses clients. Dans ce domaine, il existe peu ou pas encore de lois pour la protection du public. Les programmes universitaires sont rarement accrédités et un curriculum normatif en génie logiciel n'est qu'en voie de définition. Avant d'aborder chacun de ces problèmes, nous allons identifier ce qu'est le génie.

M. Walter Vincenti nous présente dans son livre *What Engineers Know and How They Know It* une définition du génie conçue par un ingénieur britannique, M. Rogers. Selon cette définition, «le génie se réfère aux pratiques d'organisation de la conception, de la production de n'importe quel artifice qui transforme le monde physique autour de nous afin de rencontrer des besoins reconnus» (Vincenti, 1990). Vincenti ajoute aux aspects de conception et de production du génie celui du fonctionnement de ces artifices.

En nous basant sur cette définition, nous pourrions nous demander si le génie logiciel est en fait du génie. Nous pensons personnellement que oui mais que, par contre, le génie logiciel n'est qu'à ses débuts en tant que discipline du génie. Cependant, les membres des autres disciplines du génie ne sont pas de cet avis et, jusqu'à aujourd'hui, ne considèrent pas le génie logiciel en tant que génie (communications privées avec Leonard Tripp, Président, 1999, IEEE Computer Society). Le génie logiciel doit encore beaucoup évoluer pour devenir une branche du génie. Nous entreprenons cette

recherche, en espérant qu'elle permettra au génie logiciel de progresser, même légèrement, dans son évolution. La plupart des disciplines du génie (génie civil, chimique, mécanique et autres) se sont développées au cours des dix-huitième et dix-neuvième siècles. Leur transformation en discipline solide a mis près de deux siècles (Shaw, 1990). Le génie logiciel n'existe que depuis une trentaine d'années. Laissons lui à son tour le temps de se définir comme discipline du génie.

Vu son jeune âge, le génie logiciel n'est pas un domaine bien établi. De ce fait découlent plusieurs problèmes. Nous avons mentionné ci-dessus la question à savoir si le génie logiciel fait réellement partie du génie. Ceci nous amène également à nous interroger sur ce qu'est précisément le génie logiciel. Quelles sont les activités qui en font partie? Quelles sont les qualifications requises pour être ingénieur en logiciel? Puisque la mesure joue habituellement un rôle important dans le génie (Vincenti, 1990; Kirby *et al*., 1990), les questions suivantes sont aussi pertinentes et nous intéressent particulièrement. Quelle est la place de la mesure au sein des principes fondamentaux du génie logiciel? La mesure joue-t-elle le même rôle en génie logiciel que dans les autres branches du génie? S'entend-on sur l'importance de la mesure dans le génie logiciel? Y a-t-il des problèmes de mise en oeuvre de programmes de mesure en génie logiciel? Existe-t-il des mésententes quant à ce qui doit être mesuré, aux outils de mesure, et à la validité de ces outils?

En génie logiciel, l'utilisation de la mesure est essentielle, mais pourtant sa concrétisation n'est pas toujours si simple. En effet, dans de très nombreux cas, la mise en place de programmes de mesure aboutit à un échec (Laframboise, 1996). Laframboise identifie toute une série de risques d'échec liés au contexte, aux intrants, aux processus et aux résultats de ces programmes de mesure.

La mise en place de programmes de mesure n'est pas le seul problème lié à l'utilisation de la mesure en génie logiciel. Le nombre élevé de métriques de logiciels vient aussi compliquer la situation. Il est difficile de savoir quoi mesurer en génie logiciel, quel outil utiliser pour les mesures et de vérifier si ces mesures sont adéquates. Dans leur article daté de juin 1997, Jacquet et Abran mentionnent que «de nombreuses métriques de logiciel ont été basées sur une approche intuitive non vérifiée et qu'elles ne sont pas basées sur des fondements vérifiables. Donc, un nombre important de «métriques de logiciel» ne saurait être qualifié de méthodes de mesure» (Jacquet et Abran, 1997). La validation des métriques étant essentielle, il est important d'utiliser des méthodes précises de validation de celles-ci. Certains problèmes apparaissent également à ce niveau. En effet, les auteurs de ces méthodes ne s'entendent pas sur la définition de l'expression «validation de métriques». De plus, ces auteurs

n'ont pas situé leurs propositions de validation par rapport à l'étape du processus de mesure qu'elles adressent (Jacquet, Abran et Dupuis, 1997).

L'absence de principes fondamentaux du génie logiciel retient également notre attention. Bien sûr, plusieurs auteurs ont énoncé des principes du génie logiciel. Mais, il n'existe pas un consensus selon lequel les membres importants de la communauté du génie logiciel s'entendent pour formuler les principes fondamentaux de cette discipline.

De nombreuses conséquences découlent de cette lacune. L'une d'entre elles concerne les normes. En effet, en génie logiciel, on retrouve un très grand nombre de normes, souvent incohérentes entre elles et aussi sans fondement. Dans le document «A Search for Fundamental Principles of Software Engineering», le problème des normes est décrit comme suit:

> Il est généralement énoncé que les normes de pratiques devraient être basées sur l'observation, le recensement et la validation consensuelle des meilleures pratiques implantées. Cette stratégie a cependant abouti au développement d'un corpus de normes qui sont parfois vouées à être isolées, déconnectées et désintégrées, ceci parce que chaque norme effectue une optimisation locale d'une unique pratique observable (Jabir, 1998).

Pour mieux faire ressortir le lien entre les normes et les principes fondamentaux, nous vous présentons la figure 1.1, tirée du document «A Search for Fundamental Principles of Software Engineering» (Jabir, 1998).

Comme nous montre la figure 1.1, le but recherché est d'aboutir à des principes qui organisent, expliquent et valident les normes. Ceux-ci devraient être des abstractions de ces normes. C'est pourquoi, s'il faut réussir à mettre de l'ordre dans les normes existantes du génie logiciel, il est important d'arriver à un consensus concernant les principes fondamentaux du génie logiciel.

Actuellement, les normes ne sont pas basées sur des observations empiriques. Elles résultent plutôt de travaux d'individus qui se rassemblent et forment des groupes de travail pour l'établissement de normes. C'est pourquoi on y retrouve de la duplication, des inconsistances, du chevauchement et des omissions. Une norme est peut-être une bonne représentation d'une pratique unique, mais ses liens avec les autres pratiques ne sont pas toujours évidents. Cette défaillance est due à l'absence d'une structure solide à la base, permettant de définir un certain nombre de normes fondées et cohérentes

entre elles. Les principes fondamentaux du génie logiciel pourraient peut-être constituer cette structure.

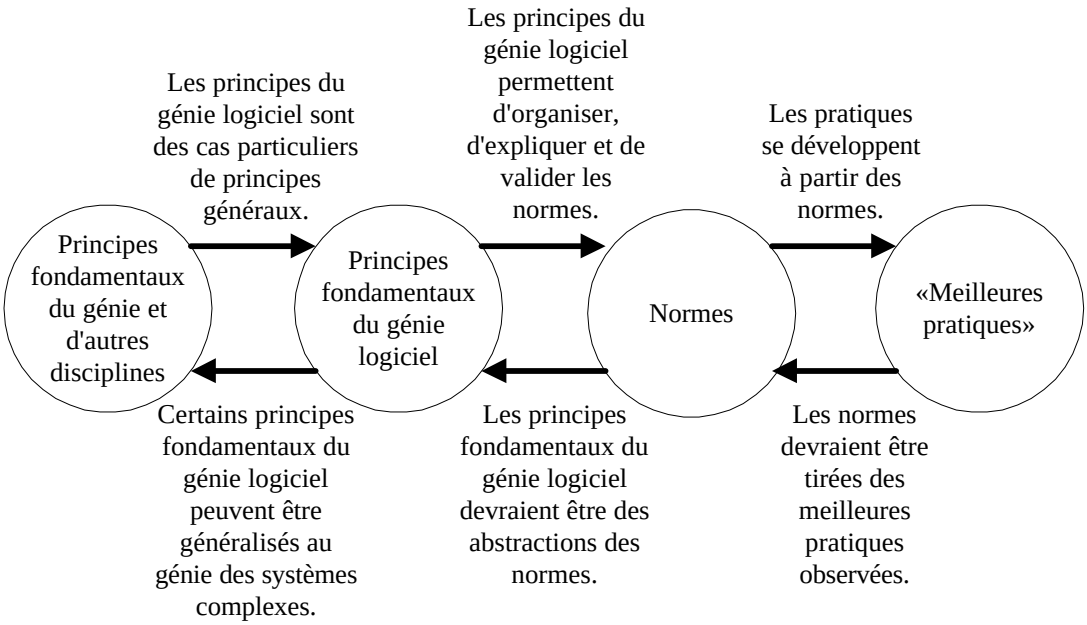


Figure 1.1 Place des principes fondamentaux (Tirée de Jabir, 1998)

Comme pour toute discipline, l'établissement de normes bien définies est essentiel en génie logiciel, car elles permettent d'apporter de la consistance et de la cohérence au travail de nombreuses personnes (Coallier et Robillard, 1985).

L'absence de principes fondamentaux influence également la reconnaissance de la discipline du génie logiciel. En effet, le génie logiciel n'est pas reconnu officiellement comme une branche du génie. Aux États-Unis et au Canada, plusieurs poursuites ont été intentées contre des firmes qui utilisaient le mot «ingénieur» dans le titre de leurs employés (Dakin, 1997).

Le génie logiciel étant très jeune, il est normal qu'il soit confronté à certains problèmes. Cependant, il serait nécessaire que les informaticiens et les ingénieurs se mettent d'accord pour préciser ce qu'est le génie logiciel. M. Parnas parle de mariage entre ces deux branches. Il dit ceci:

> Le génie logiciel est souvent traité comme une branche de l'informatique. C'est comme considérer le génie chimique comme une branche de la chimie. Nous avons besoin à la fois des chimistes et des ingénieurs chimiques, mais l'un et l'autre sont très différents. Les

chimistes sont des scientifiques; les ingénieurs chimiques sont des ingénieurs. Le génie logiciel et l'informatique possèdent le même genre de relation (Parnas, 1997).

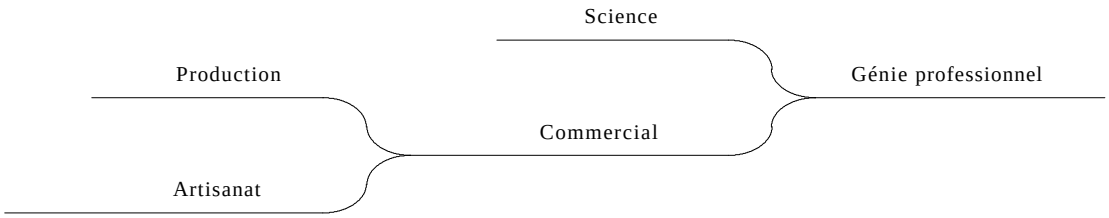


Figure 1.2 Évolution d'une discipline du génie (Tiré de Shaw, 1990)

Dans son article intitulé «Prospects for an engineering discipline of software», Mary Shaw de l'université Carnegie Mellon montre l'évolution d'une discipline vers le génie (voir figure 1.2).

Les lignes inférieures montrent le parcours que suit la technologie; tandis que les lignes supérieures montrent l'entrée des habiletés de production et de la connaissance scientifique, contribuant ainsi à de nouvelles possibilités au niveau des pratiques d'ingénierie. Selon Shaw, le génie logiciel serait entre l'artisanat et le commercial, selon les aspects dont on tient compte. Elle pense que la définition de la discipline du génie logiciel constitue un objectif pouvant être atteint, mais que certaines conditions sont requises. Elle exprime ces conditions comme suit: «Sélectionner un ensemble approprié de contributions purement empiriques, à court terme, pragmatiques, qui aideront à stabiliser les pratiques commerciales et à investir dans des efforts à long terme pour développer et rendre disponible des contributions scientifiques de base (Shaw, 1990).

C'est ici que l'on peut faire le lien avec l'établissement de la discipline du génie logiciel et la nécessité d'identifier les principes fondamentaux de cette discipline. En effet, si l'on se base sur la

figure 1.1 de ce document, les principes fondamentaux du génie logiciel pourraient représenter les meilleures pratiques du domaine, lesquelles permettraient d'établir les normes pour la commercialisation de logiciels. D'autre part, l'obtention de ces principes nous permettrait d'évaluer la place de la mesure en génie logiciel.

La question que nous nous posons est la suivante: quelle est la place de la mesure au sein des principes fondamentaux du génie logiciel? Cette question préoccupe de nombreuses personnes dont les activités sont liées d'une façon ou d'une autre au génie logiciel. C'est pourquoi, certains membres

du Laboratoire de recherche en gestion de logiciels de l'Université du Québec à Montréal s'y sont intéressés. Ce laboratoire se spécialise en études dans le domaine du génie logiciel. Les intérêts des chercheurs incluent les métriques de logiciel, dont particulièrement l'analyse des points de fonction, la mesure de la maintenance, les méthodes de validation de mesures et les programmes de mesure. Dans le cadre des recherches du laboratoire, des travaux ont été effectués, tentant d'identifier les principes fondamentaux du génie logiciel. Ces travaux ont permis d'obtenir une liste de principes importants. Les participants sont arrivés à s'entendre sur le caractère fondamental de certains de ces principes. Notre recherche constitue le prolongement de ces travaux et s'inscrit parmi les activités en cours au Laboratoire de recherche en gestion de logiciels. Nous pensons que cette recherche peut donner lieu à plusieurs recherches subséquentes dans les domaines du génie logiciel et de la mesure. Notre recherche fait notamment partie d'un projet plus général consistant à produire un guide sur le corpus de connaissances en génie logiciel. Ce projet a été assigné à l'Université du Québec à Montréal par les responsables de la IEEE Computer Society.

L'obtention de principes fondamentaux du génie logiciel nous permettrait d'évaluer la place de la mesure dans le génie logiciel. En effet, nous savons que les autres types de génie (génie civil, électrique, aéronautique, ou autres) utilisent énormément de mesures. Par contre, en génie logiciel, nous ignorons la place que l'on accorde à la mesure. Des problèmes tels que ceux de la mise en oeuvre de programmes de mesure, de savoir quoi mesurer, quand et pourquoi et quel outil utiliser pour le faire, ainsi que les problèmes de validité de ces outils pourraient être plus facilement résolus si la place de la mesure en génie logiciel était clairement déterminée.

Ayant défini le problème qui nous préoccupe ici, nous nous proposons de préciser, dans la prochaine section, les objectifs de notre démarche.

2. OBJECTIFS

L'objectif principal de notre recherche est de produire un questionnaire sur la place de la mesure en génie logiciel. Ce questionnaire pourrait ensuite être utilisé lors de recherches subséquentes afin d'étudier la place de la mesure en génie logiciel. Nous nous intéressons à l'aspect mesure en génie logiciel car, pour les autres disciplines du génie, la maturité est souvent associée à la qualité des mesures qu'on sait y prendre. Notre but serait de faciliter l'évaluation future de la place de la mesure au sein du génie logiciel.

Afin d'arriver à produire ce questionnaire, il nous faut identifier les incertitudes concernant la mesure en génie logiciel. Ceci constituera un sous-objectif de notre recherche. Plusieurs aspects pouvant être abordés sont le quoi, le quand, le comment (procédures et outils), le pourquoi (dans quels buts) de la mesure en génie logiciel. Nous nous intéresserons particulièrement au pourquoi, mais nous aborderons également le comment et le quoi de la mesure en génie logiciel. Nous faisons le choix de nous attarder au pourquoi principalement car il nous paraît logique d'identifier les objectifs de la mesure avant de se préoccuper du comment mesurer.

Pour atteindre ces objectifs, nous avons pensé utiliser les résultats de recherches déjà réalisées sur les principes fondamentaux du génie logiciel. L'analyse de la liste de ces principes, les commentaires faits par les participants (membres des comités IEEE et praticiens du génie logiciel) à deux études Delphi et à un sondage ainsi que la revue de littérature permettraient d'identifier les incertitudes de la mesure en génie logiciel et, à partir de là, de produire un questionnaire. Il nous faudra alors analyser les principes fondamentaux obtenus lors des deux études Delphi, ainsi que les commentaires qui accompagnaient ces principes. Ce que nous appelons les commentaires des études Delphi sont les remarques faites par les participants à ces études au sujet de notre liste de principes candidats. Ils peuvent être négatifs, positifs, critiques, représenter une opinion, dénoncer un chevauchement de principes, etc.

OBJECTIFS PERSONNELS

Ces objectifs concernent l'obtention d'une meilleure connaissance du génie logiciel et de l'utilisation qu'on y fait de la mesure. Un autre objectif personnel est de contribuer, tant soit peu, à la définition du génie logiciel en tant que branche du génie. Quoique minime, nous espérons que notre démarche

sera à l'origine d'autres démarches vers l'établissement de cette discipline. Espérant faire carrière dans le domaine du génie logiciel, nous désirons ainsi accroître nos connaissances du domaine.

3. MÉTHODES UTILISÉES POUR ABOUTIR À NOS OBJECTIFS

3.1 CADRE CONCEPTUEL

Une nouveau domaine, celui du génie logiciel, est en pleine émergence. Malgré ce que laisse croire son nom, il n'est pas encore officiellement reconnu comme branche du génie par les membres des autres disciplines du génie, bien que l'état du Texas aux États-Unis s'apprête à le faire en juin 1998 (Bagert, 1998). Notre recherche s'insère dans le développement de cette nouvelle discipline. L'aspect qui nous intéresse particulièrement est celui de la place de la mesure au sein du génie logiciel. Notre but est donc de produire, à l'aide des commentaires fournis par les participants aux études Delphi (voir section 3.3), du sondage et de la revue de littérature, un questionnaire qui permettrait d'analyser la vision de personnes du domaine sur la place de la mesure à l'intérieur des principes fondamentaux du génie logiciel.

Dans les paragraphes qui suivent, nous vous présenterons le résultat de nos recherches sur l'utilisation de la mesure dans le génie tout au long de son histoire et de son rôle au sein de ces différentes disciplines. Nous tenterons de vous démontrer ainsi l'importance de la mesure en génie. Nous présenterons également des exemples d'utilisation de la mesure en génie.

LA MESURE AU COURS DE L'HISTOIRE DU GÉNIE ET DANS CERTAINES BRANCHES SPÉCIFIQUES DU GÉNIE

En nous rapportant au livre «Engineering In History» (Kirby *et al*., 1990), nous nous rendons compte que Kirby mentionne l'utilisation de mesures pour accomplir les grands événements de cette histoire. En parlant des Égyptiens, les auteurs nous disent que «de placer un obélisque, pesant plusieurs centaines de tonnes, à la verticale est un fait important du génie qui nécessite des calculs précis et un équipement spécial, même dans les temps modernes» (Kirby *et al*., 1990).

Quelques siècles plus tard, les Grecs et les Romains ont également participé à l'avancement du génie. Les Grecs ont surtout contribué au génie de par l'utilisation de principes mathématiques pour effectuer les calculs. Les auteurs prétendent que les contributions d'Archimède, c'est-à-dire la gravité spécifique, la pression des liquides et les principes de l'action des leviers, sont fondamentaux pour les

ingénieurs aujourd'hui. À la même époque, les Romains ont contribué à l'utilisation d'instruments de mesure tels le *groma* ou le *libella*.

Les contributions des ingénieurs aux seizième et dix-septième siècles ne sont pas non plus négligeables. Les auteurs avancent que «Parmi plusieurs autres noms, on peut citer ceux de Napier, Briggs, Oughtred, Descartes, Pascal, Newton, Leibniz, qui ont tous contribué au système complexe de calculs duquel dépend le génie moderne» (Kirby *et al*., 1990).

Au dix-huitième siècle, la mesure a joué un grand rôle dans l'amélioration de technologies telles les roues à eau, les routes, les canaux et les ponts, les bateaux à vapeur, les locomotives et la construction. Plus récemment, des personnalités importantes, parmi lesquelles ont peut citer Coulomb, Ampère, Faraday, Ohm, Oersted et Edison, ont grandement participé à l'établissement des disciplines du génie électrique et de l'électromagnétisme. La mesure tenait une place importante dans l'effort d'innovation entrepris par ces chercheurs.

En consultant le livre intitulé «Origins of Industrial Engineering» (Emerson et Douglas, 1988), nous avons appris que le génie industriel est basé principalement sur l'utilisation de deux types de mesures: l'étude du temps et l'étude du mouvement. Taylor est considéré comme le pionnier de cette discipline. En travaillant dans une manufacture, il avait eu l'idée d'améliorer les méthodes de travail des employés, ceci, dans le but de diminuer leurs charges et leurs heures de travail. Un exemple de ces études est par exemple, la mesure de l'activité de pelletage, où il en était venu à la conclusion que les travailleurs avaient besoin de pelles de dimensions différentes pour chaque tâche, selon la densité des matériaux qu'ils devaient pelleter. Les auteurs décrivent également une technique inhérente au génie industriel intitulée: Conception et mesure du travail. Nous citons la description de cette technique.

> La conception et la mesure du travail est une technique de base du génie industriel. Elle implique la mesure du travail, de sorte que des attentes de travail, devant être complétés en un laps de temps**,** ou des standards puissent être déterminés. Puis, en utilisant ces données, des horaires de travail journaliers peuvent être conçus à partir des horaires de production (Emerson et Douglas, 1988).

Dans le livre «What Engineers Know and How They Know It: Analytical Studies from Aeronautical History», M. Vincenti (1990) cherche à nous expliquer en quoi consiste les activités d'un ingénieur et aussi ce qui différencie un ingénieur d'un scientifique. Pour ce faire, il se base sur des exemples tirés

de son domaine, le génie aéronautique, bien qu'il affirme que ce qu'il dit concerne le génie en général.

Le premier exemple que nous donne Vincenti est celui de la conception du profil d'une aile d'avion. Le problème consiste à déterminer les formes et dimensions permettant les meilleures performances aérodynamiques. Pour cela, toutes sortes de tests sont effectués à plusieurs endroits. L'utilisation d'un tunnel de vent permet d'expérimenter les différents types de profils. Plusieurs formules sont également établies. Cependant, les ingénieurs aéronautiques mettront des décennies à maîtriser la question.

Vincenti nous explique également comment il a fallu définir des mesures précises permettant l'évaluation des qualités de vol d'un avion. Ces mesures étaient devenues indispensables car les descriptions des qualités de vol d'un avion par les pilotes étaient trop floues pour permettre aux ingénieurs d'effectuer les ajustements nécessaires. Ces problèmes de définition de la qualité de vol rappellent ceux auxquels on doit souvent faire face en génie logiciel lors de la définition des exigences d'un système d'information.

Un autre aspect intéressant de l'utilisation de la mesure en génie aéronautique concerne la stabilité versus la mobilité d'un avion. Le problème se posait comme suit: l'augmentation de la stabilité d'un avion diminuait sa mobilité et vice-versa. Il s'est donc avéré nécessaire pour les ingénieurs de calculer le meilleur ratio stabilité/mobilité qu'un avion devrait possédait. Cela a occasionné bien des calculs.

Les exemples mentionnés ci-dessus nous permettent de prendre conscience de l'importance de la mesure en génie. En génie logiciel, la mesure est également très importante. Dans la prochaine section, nous allons voir les raisons qui portent à effectuer des mesures dans ce domaine.

POURQUOI MESURER EN GÉNIE LOGICIEL

Dans leur livre intitulé *Applying Software Metrics*, Oman et Pfleeger (1994) identifient six raisons pour lesquelles il faudrait mesurer en génie logiciel. Ce sont les suivantes:

- Mesurer pour la compréhension: certaines mesures nous permettent de mieux comprendre les activités de développement et de maintenance de logiciels. Nous pouvons ainsi évaluer la

situation courante en établissant des points de base qui nous aident à fixer les objectifs pour les comportements futurs.

- Mesurer pour l'expérimentation: l'expérimentation est nécessaire en génie logiciel afin d'améliorer notre connaissance de la façon de développer des logiciels, de mieux comprendre les effets des technologies diverses et d'identifier les champs nécessitant le plus d'amélioration. Les mesures jouent un rôle important dans l'expérimentation, nous permettant de tester des hypothèses et d'en bâtir de nouvelles.
- Mesurer pour le contrôle de projet: les mesures nous permettent d'évaluer où nous sommes rendus dans un projet et de prédire ce qui se produira dans le futur. Elles nous permettent ainsi de rectifier le tir, si nécessaire, afin d'atteindre nos objectifs.
- Mesurer pour l'amélioration du processus: la mesure nous permet de mieux évaluer le processus, de mieux comprendre la portée des changements qui lui sont apportés et ainsi de l'améliorer.
- Mesurer pour l'amélioration du produit: la mesure offre une vision sur les façons dont les processus, les produits, les ressources, les méthodes et les technologies de développement de logiciels s'apparentent. Les mesures peuvent nous aider à répondre à des questions sur l'efficacité des techniques et des outils, la productivité des activités de développement, la qualité des produits et plus. Ces données nous permettent d'améliorer le produit. (ISO/IEC 9126:1991).
- Mesurer pour faire des prévisions: pour de nouvelles activités, il faut être à même de prévoir l'effort requis, ainsi que les coûts de développement et autres. Les mesures nous permettent d'avoir un point de repère afin de faire des prévisions sur ces activités. Le fait d'attendre simplement la fin du projet, et ensuite de mesurer les attributs de coût et de temps, est clairement inacceptable.

Dans le livre «Software Metrics: A Rigorous & Practical Approach» (Fenton et Pfleeger, 1997), nous avons identifié une septième raison de mesurer. Elle s'apparente un peu à la mesure pour faire des prévisions, mais se concentre surtout sur le passé et le présent, plutôt que sur le futur. C'est la mesure pour l'évaluation:

- Mesurer pour l'évaluation: Mesurer pour l'évaluation est très utile afin de pouvoir comprendre ce qui existe maintenant ou ce qui s'est déroulé dans le passé.

Ces catégories de mesure vont nous servir de grille d'évaluation pour notre évaluation du rôle de la mesure en génie logiciel.

3.2 DÉMARCHES DEVANT ÊTRE EFFECTUÉES AFIN D'ATTEINDRE LES OBJECTIFS DE LA RECHERCHE

Dans la présente section, nous abordons les différentes méthodes que nous utiliserons pour notre recherche. Nous vous expliquons chacune de ces méthodes en particulier, mais nous tenons d'abord à vous donner une vue globale des démarches que nous entreprendrons afin de nous permettre d'atteindre les objectifs que nous nous sommes fixé. Rappelons que notre objectif principal est d'évaluer la place de la mesure en génie logiciel en utilisant les principes fondamentaux.

Nous commençons notre recherche en effectuant notre collecte de données. La collecte de données consiste dans les résultats de deux études Delphi ainsi que ceux du sondage effectué auprès des praticiens du génie logiciel.

Parallèlement à notre cueillette de données, nous procédons à une revue de la littérature afin de mieux connaître les domaines que nous abordons c'est-à-dire ceux du génie, du génie logiciel, des principes fondamentaux et de la mesure. Pour plus d'informations sur les livres que nous consulterons, voir la section 3.3.1 du présent document, portant sur la revue de la littérature.

Dans la dernière étape de notre recherche, nous utiliserons les résultats des deux études et du sondage ainsi que des connaissances acquises de part la revue de littérature pour identifier les incertitudes de la mesure en génie logiciel. Les commentaires se rapportant à la mesure seront classés d'après les différentes raisons de mesurer identifiées en génie logiciel (Oman et Pfleeger, 1997; Fenton et Pfleeger, 1997). Puis, une description de l'analyse de la place de la mesure en génie logiciel sera effectuée. À partir de cette analyse, nous produirons un questionnaire qui pourrait permettre d'analyser la place de la mesure au sein des principes fondamentaux du génie logiciel. Les détails de l'analyse des données qualitatives se trouvent à la page 23 de ce document.

La figure 3.1 illustre les différentes étapes ayant précédé notre recherche. Cette figure est tirée de l'article *Principes fondamentaux du génie logiciel: Une étude Delphi* (Dupuis *et al*., 1997). La figure 3.2 illustre le contexte dans lequel cette activité de synthèse se situe.

Au tableau D.1 en annexe, nous présentons le cadre de Basili modifié qui nous permet de donner une représentation synthétique de ces démarches.

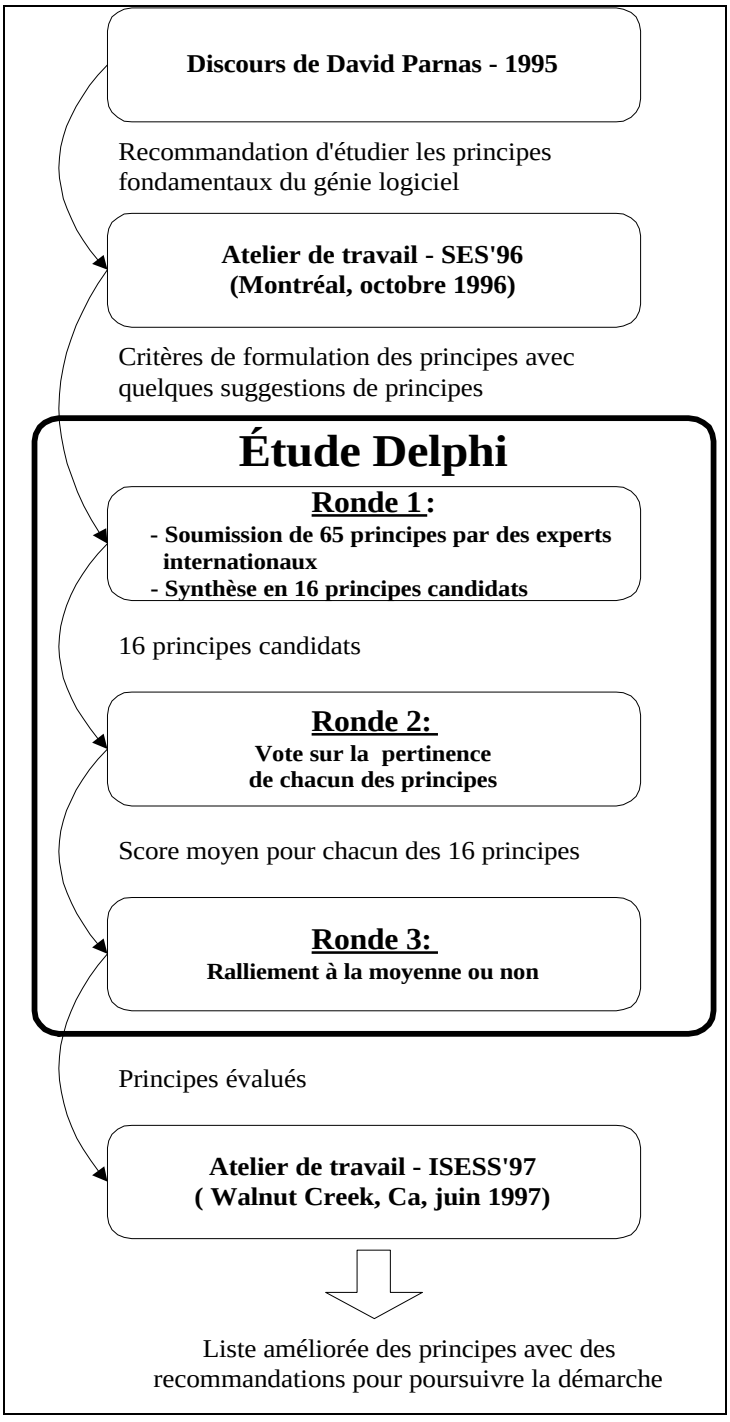


Figure 3.1 Travaux effectués sur l'identification des principes fondamentaux du génie logiciel (Tirée de Dupuis *et al*., 1997)

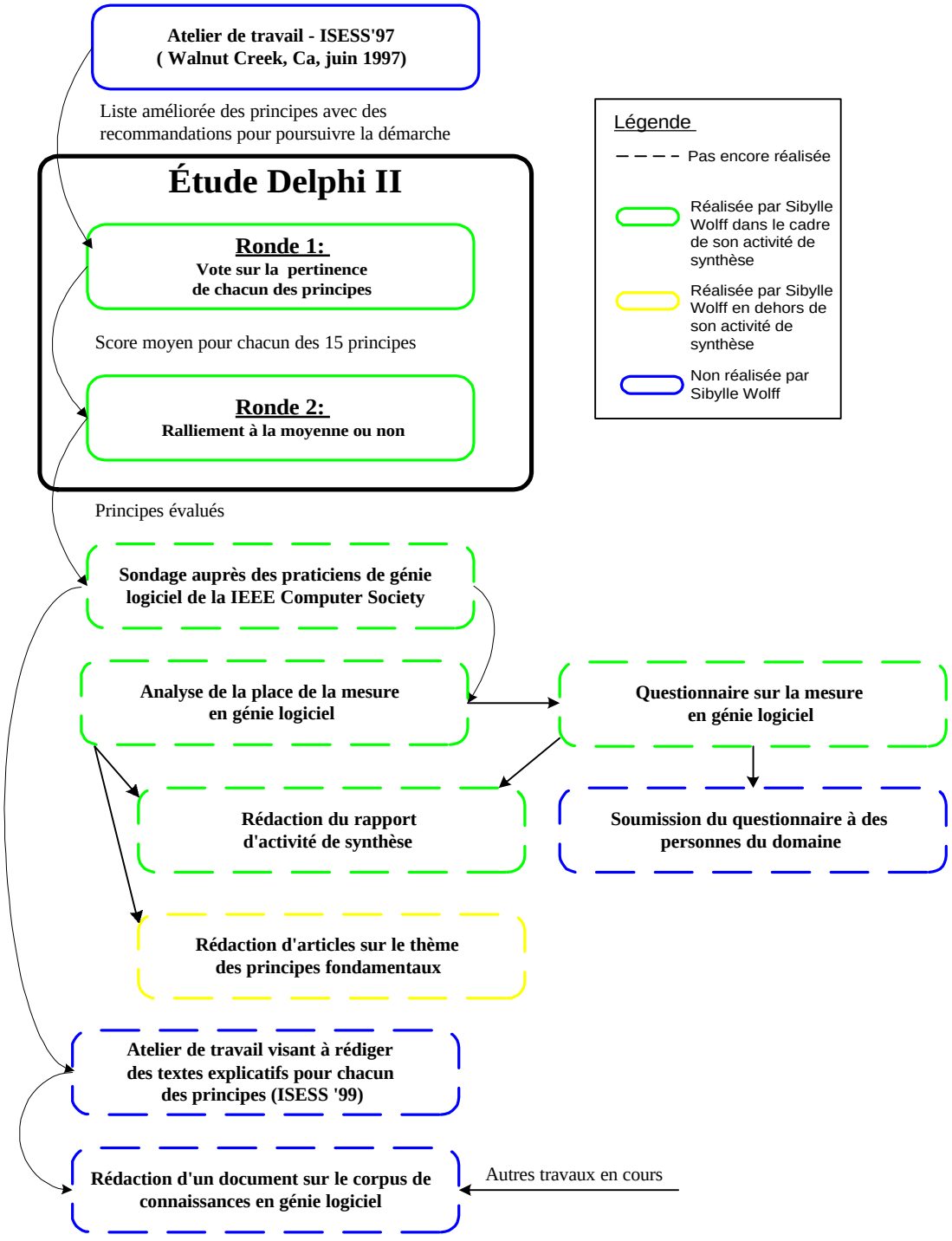


Figure 3.2 Autres travaux de recherche sur les principes fondamentaux du génie logiciel

3.3 MÉTHODE DE CUEILLETTE DE DONNÉES

Afin d'atteindre les objectifs que nous nous sommes fixé pour notre recherche, nous devons utiliser un certain nombre de données déjà recueillies. Nos principales sources de données secondaires sont une revue de la littérature et l'utilisation des résultats d'une étude Delphi et de quelques ateliers qui ont déjà eu lieu dans le but d'identifier les principes fondamentaux du génie logiciel. De plus, pour compléter nos données, nous prévoyons également de recueillir des données primaires, par le biais d'une deuxième étude Delphi et d'un sondage. Nous vous présentons ci-dessous les détails de ces sources de données:

3.3.1 Revue de la littérature

Dans notre revue de littérature, nous avons pour objectif de consulter certains livres ainsi que des articles de colloques et de revues. Les sujets sur lesquels nous nous concentrons sont les suivants:

1. Le génie
   Nous voulons chercher à mieux comprendre le génie et à avoir une compréhension initiale du rôle qu'y joue la mesure.
2. Le génie logiciel
   Nous voulons chercher à comprendre ce qu'est le génie logiciel actuellement ainsi que la place qu'il occupe au sein du génie.
3. Les principes fondamentaux du génie logiciel
   Nous voulons analyser les principes se trouvant dans les textes écrits par Jabir 1997 et Dupuis *et al*. (1997) ainsi que les messages finaux des deux études Delphi (voir annexe A et B).
4. La mesure en génie logiciel
   Nous essaierons de comprendre la place de la mesure en génie logiciel.

Cette revue de littérature aura pour objectifs de nous donner une meilleure compréhension de notre sujet et de nous permettre de présenter des arguments solides. Nous nous attarderons à comprendre le rôle de la mesure dans le génie logiciel. Pour ce faire, nous utiliserons les livres et articles suivants:

- E*ngineering in History* (Kirby *et al*., 1990)
- *What Engineers Know and How They Know It* (Vincenti, 1990)
- *Origins of Industrial Engineering: The Early Years of a Profession* (Emerson et Naehring, 1988)
- «A search for Fundamental Principles of Software Engineering» (Jabir, 1998)
- «Principes fondamentaux du génie logiciel: Une étude Delphi» (Dupuis *et al*., 1997)
- «Prospects for an engineering discipline of software» (Shaw, 1990)
- «Will There Ever Be Software Engineering?» (Jackson, 1998)

- *Applying Software Metrics* (Oman et Plfeeger, 1997)
- «Software Engineering: An Unconsummated Marriage» (Parnas, 1997)
- *Software Metrics: A Rigorous & Practical Approach* (Fenton et Pfleeger, 1997)

À cette étape de notre recherche, nous identifierons les questions qui devraient être posées afin d'analyser la place de la mesure en génie logiciel, en nous basant sur les catégories d'utilisation de la mesure énumérées dans la section intitulée «POURQUOI MESURER EN GÉNIE LOGICIEL» et se trouvant à la page 12 du présent document.

3.3.2 Démarches d'identification des principes fondamentaux du génie logiciel

Lors d'un atelier de travail durant la conférence SES de 1996, les participants, ont tenté de définir des critères pour l'établissement de principes fondamentaux en génie logiciel. Ils sont arrivés à un ensemble de huit critères auxquels devaient répondre les principes (Jabir, 1998).

Une étude Delphi consiste à sélectionner un certain nombre de personnes à même de répondre à nos besoins, et à leur fournir un questionnaire à chaque étape de l'étude. Les questionnaires de chaque étape, à partir de la deuxième étape, sont basés sur les réponses des étapes précédentes. Ainsi, le texte accompagnant ces questionnaires permet d'informer les participants des réponses de leurs collègues et d'obtenir un consensus entre les participants à la fin de l'étude.

Nous avons choisi l'étude Delphi parce qu'elle se prêtait bien à nos besoins. En effet, dans une étude Delphi, les participants n'ont pas besoin de se retrouver au même endroit en même temps, ce qui élimine les contraintes de temps et d'argent. Ainsi, nous avons pu obtenir l'opinion d'experts dispersés à travers le monde, et ceci à peu de frais. L'anonymat des participants est respecté jusqu'à la fin de l'étude, permettant ainsi d'éviter la domination de certains individus durant le processus. Les différentes étapes de l'étude nous permettaient d'obtenir une liste générale de principes et de la raffiner au fur et à mesure que l'étude avançait.

HISTORIQUE

Au printemps 1997, Robert Dupuis et Pierre Bourque, deux des participants de l'atelier au SES'96 ont réalisé une étude Delphi auprès de quatorze personnalités très connues du génie logiciel. Cette étude comprenait trois tours. Au premier tour, chaque participant avait pour tâche d'énoncer cinq principes fondamentaux du génie logiciel. Dans ce cas précis, une compilation des 65 principes recueillis avait permis d'aboutir à une liste de seize principes. Au deuxième tour, les participants devaient coter chacun de ces seize principes sur une échelle de 1 à 10, où 1 signifiait qu'un individu était totalement en désaccord avec ce principe et 10 correspondait à un accord total. Entre le deuxième et le troisième tour, les moyennes des cotes obtenues pour chacun des seize principes était calculées. Au dernier tour, les participants devaient indiquer s'ils approuvaient la cote moyenne obtenue pour chaque principe. Les organisateurs de cette étude Delphi avaient décidé qu'il y aurait consensus à partir de 10 réponses positives sur 12 participants. Au dernier tour, on ne comptait que douze participants, ce qui explique le dix sur douze pour le consensus. Le résultat final de cette étude avait été un classement des seize principes en trois catégories:

1. une catégorie d'énoncés pour lesquels il y avait consensus sur le fait qu'ils constituaient des principes fondamentaux du génie logiciel
2. une catégorie pour laquelle il y avait consensus sur le fait que ces énoncés ne constituaient pas des principes fondamentaux du génie logiciel
3. une catégorie où il n'y avait pas consensus

Par la suite, un deuxième atelier s'est tenu à la conférence ISESS de juin 1997. Durant cet atelier, il y a eu des propositions de modifications et d'ajout de critères et de principes. Les résultats de cet atelier et des démarches précédentes sont listés en annexe.

Les données de ces trois étapes d'identification des principes fondamentaux du génie logiciel seront utilisées dans notre recherche. Elles serviront en particulier de point de départ pour la conduite de notre propre étude Delphi, laquelle est décrite dans la prochaine sous-section.

3.3.3 Deuxième étude Delphi

Après la présentation des résultats de la première étude Delphi, les membres d'un atelier avaient suggéré d'effectuer une étude semblable auprès des membres des comités de la IEEE Computer

Society qui sont fortement impliqués dans le génie logiciel dans le cadre de leur travail. Cette deuxième étude Delphi constitue la concrétisation de cette proposition. Elle a permis de raffiner les résultats obtenus lors de la première étude. Elle a également permis de clarifier la position des énoncés pour lesquels il n'y avait pas eu de consensus, à savoir si ce sont des principes fondamentaux du génie logiciel ou pas, ou encore, s'il est nécessaire de les reformuler.

Contrairement à la première étude, la deuxième comprenait deux tours. Ces premier et deuxième tours correspondent respectivement aux deuxième et troisième tours de la première étude. Lors du premier tour, une liste de principes a été fournie aux participants ainsi qu'une liste de critères que doivent respecter ces principes. Comme nous l'avons déjà mentionné, ces principes et critères nous proviennent des démarches précédentes d'identification des principes fondamentaux du génie logiciel décrites plus haut. Les participants ont eu à noter, sur une échelle de 1 à 10, les différents principes proposés, selon l'importance qu'ils leur accordent en génie logiciel. Les participants ont ensuite retourné les résultats aux organisateurs de l'étude Delphi. Ces derniers ont compilé les données en calculant la médiane de chaque principe et en recueillant les commentaires émis par les participants. Au deuxième tour, les principes ont été à nouveau soumis aux participants, mais cette fois dans le but de leur faire dire s'ils approuvaient ou non la moyenne obtenue par chaque principe au premier tour. Chaque participant a également reçu les différents commentaires des autres participants, ce qui pouvait lui permettre de se rallier ou de s'opposer à un autre point de vue.

Il est bon de mentionner qu'une étude Delphi se fait dans l'anonymat, ceci jusqu'à l'obtention du résultat final. Ceci évite l'influence de certains participants sur d'autres. Cependant, lorsque l'étude Delphi est terminée, il n'est plus indispensable de garder l'anonymat puisque les opinions sont déjà émises. Dans notre cas précis, nous avons demandé aux participants à la fin du deuxième tour, s'ils acceptaient de révéler leur identité. L'identité des personnes qui ont accepté que leur nom soit utilisé a été révélée en même temps que les résultats à tous les participants à la fin de l'étude. Cependant, les commentaires sont restés impersonnels.

Les résultats des deux études Delphi consistent en une cote médiane pour chacun des principes, le nombre de personnes acceptant ces énoncés comme principes fondamentaux du génie logiciel et une série de commentaires sur les principes fondamentaux, fournis par les participants. Les commentaires sont pour nous d'une très grande richesse car ils nous permettent d'éliminer, de rajouter ou de modifier les principes candidats. Ce travail de raffinement de notre liste de principes

candidats au cours des différentes démarches de recherche sur le sujet nous permettra un jour de présenter une liste de ceux que nous pensons être des principes fondamentaux.

PRINCIPES FONDAMENTAUX DU GÉNIE LOGICIEL

Dans cette section, nous vous présentons la liste de quinze principes fondamentaux du génie logiciel, ainsi qu'un court texte accompagnant chacun de ces principes. Ce texte constitue une synthèse des commentaires faits par les participants pour chacun des principes lors des études Delphi. Le principe A est relié à la mesure. Nous nous attendons à ce que les commentaires s'y rapportant soient pertinents à la mesure et nous permettront ainsi de décrire les incertitudes concernant la mesure en génie logiciel et de produire le questionnaire. D'autres principes n'impliquent pas directement la mesure mais nous présumons que certains commentaires s'y rapportant pourraient également nous intéresser. On peut citer, par exemple, les principes E et F où il est question de rigueur et de formalisme.

A. Appliquer et utiliser des mesures quantitatives dans la prise de décision: la prise de décision doit être basée sur des données quantitatives pour le choix d'outils, aussi bien que pour le choix de méthodologies et de gestion de projets. Une méthode scientifique doit être utilisée. Les résultats doivent être prédits là où c'est attendu, et les différences avec les résultats actuels doivent être utilisées comme guides pour la gestion.
B. Construire en réutilisant et pour être réutilisé: dès que c'est possible, tentez de réutiliser les patterns existants. Produisez des logiciels modulaires et indépendants.
C. Contrôler la complexité de par multiples perspectives et multiples niveaux: le logiciel est intangible de nature et est l'un des artefacts les plus complexes bâtis par l'humanité. Des points de vue variés et de multiples niveaux de synthèse sont donc requis afin de saisir correctement cet artefact et de gérer cette complexité inhérente.
D. Définir les artefacts logiciels de façon rigoureuse: appliquez de la rigueur et efforcez vous d'obtenir des patterns prévisibles pour les données, le programme, l'architecture, les interfaces et la logique procédurale afin de maintenir la cohérence et de réduire l'entropie des processus et des produits.
E. Mettre en place un processus logiciel flexible: faites en sorte que les propriétés secondaires puissent être facilement modifiées. Pour accomplir ceci, incorporez dans le plan de projet des points de revue qui permettent d'effectuer des changements au sein des exigences.
F. Implanter un processus rigoureux et l'améliorer continuellement: la qualité des processus utilisés pour développer des logiciels déterminera largement la qualité, la durabilité et l'efficacité de coût du résultat.
G. Consacrer les ressources nécessaires à la compréhension du problème: les produits logiciels sont, de façon inhérente, mal définis. Conséquemment, assurez vous de bien saisir le domaine d'application. Travaillez avec le client pour obtenir une bonne compréhension du domaine. Identifiez et testez les suppositions le plus tôt possible.
H. Gérer d'une manière aussi formelle que possible la qualité durant l'ensemble du cycle de vie: la qualité devrait être encastrée dans les produits et processus. Construisez la solution de sorte

qu'elle puisse être évaluée avant qu'aucun code ne soit généré et établissez des objectifs qui pourront être utilisés pour évaluer le produit tout au long de l'étape de développement.

I. Minimiser les interactions entre les composants logiciels: construisez des logiciels simples et modulaires, en utilisant des architectures transparentes et en évitant toutes innovations et complexité inutiles.

J. Produire le logiciel par étapes: commencez à partir d'un noyau qui peut être utile à l'utilisateur, et construisez successivement des approximations plus proches du produit fini.

K. Établir des objectifs de qualité pour chaque produit à livrer: des objectifs doivent être définis pour chaque produit livrable puisque le même niveau de qualité ne peut être accepté pour tous. L'attention doit être mise sur les fonctions les plus utilisées et/ou les plus critiques.

L. Comme le logiciel est, de par sa nature même, sujet au changement, il faut planifier et gérer ce changement: incorporez un plan de focus sur la maintenance dans le plan de projet. Produisez des logiciels qui se décrivent d'eux-mêmes. Identifiez les parties changeant fréquemment et documentez uniquement ce qui est nécessaire.

M. Les compromis étant inhérents au génie logiciel, il faut les rendre explicites et les documenter: puisqu'un nombre arbitraire de bons buts arbitraires ne peut être atteint, l'utilisation attendue devrait être quantifiée. Des alternatives de conception devraient être explorées. Un choix explicite devrait être fait entre elles, afin de distinguer les propriétés essentielles des propriétés secondaires.

N. Pour améliorer la conception, étudier les solutions antérieures à des problèmes similaires: pour améliorer nos chances de capturer les exigences inconnues et afin de profiter des leçons apprises, étudiez les solutions antérieures à des problèmes similaires.

O. L'incertitude est inévitable en génie logiciel. Il faut identifier et gérer cette incertitude: l'incertitude doit être acceptée. Les risques doivent être reconnus et des plans d'urgence doivent être incorporés dans la planification des projets.

3.4 MÉTHODE DE TRAITEMENT DES DONNÉES

Nous voulons appliquer des tests statistiques très simples, tels la médiane et l'écart-type, sur les résultats de notre étude Delphi. Nous tenterons également de faire ressortir les différents *patterns* qui pourraient exister dans nos données.

Nous avons mentionné plus haut que nous chercherions les corrélations entre les résultats de nos études et un sondage effectué auprès de praticiens du génie logiciel et membres de la IEEE Computer Society. Pour ce faire, nous utiliserons des méthodes statistiques pour faire ressortir les corrélations. Le sondage nous permettra également de récolter des données sur les opinions des praticiens du génie logiciel, en analysant des données sur le pays, le nombre d'années d'expérience, le nombre d'années de pratiques dans le domaine du génie logiciel, le niveau d'études atteint, le(s) domaine(s) d'étude. Ce sondage devrait être envoyé à quelques milliers de praticiens en génie logiciel de l'IEEE. Deux sondages effectués au sein de la IEEE Computer Society ont obtenu un taux de réponse d'environ 17%. Afin de nous assurer de la qualité de notre sondage, nous avons établi un processus de vérification qui consistait en une révision du sondage à trois niveaux différents. À un premier niveau, le sondage a été vérifié par trois personnes. Un groupe de 10 personnes a procédé à une seconde vérification. Avant d'être envoyé à ces quelques milliers de praticiens, le sondage sera envoyé à une centaine de praticiens présentant les mêmes caractéristiques que les autres. À la suite de chacune de ces vérifications, des changements ont été ou seront effectués, dans le but d'améliorer notre questionnaire. Ce questionnaire vous est présenté à l'annexe C de ce document.

MÉTHODE D'ANALYSE DE DONNÉES QUALITATIVES

Pour définir la démarche d'analyse que nous utiliserons dans notre recherche, nous nous sommes basés sur le livre «Analyse des données qualitatives» (Huberman et Miles, 1991). Les commentaires des deux études Delphi effectuées afin d'identifier les principes fondamentaux du génie logiciel constituent les données qualitatives que nous aurons à analyser.

Nous avons établi des catégories pour la classification de nos données. Ces catégories ont été déterminées lors de l'analyse préliminaire de ces données qualitatives.

Nous avons d'abord classé les commentaires en fonction du principe auquel ils se rapportaient. Puis, nous avons établi des catégories de type de commentaires. Ces catégories sont les suivantes:

- FO: Commentaires portant sur la formulation des principes
- OP: Commentaires exprimant une opinion sur un principe
- CH: Commentaires indiquant un chevauchement entre deux ou plusieurs principes
- CO: Commentaires indiquant la possibilité de combiner deux principes
- CC: Commentaires sur les commentaires accompagnant les principes
- CT: Commentaires indiquant des contradictions entre les principes
- SC: Commentaires sur le score accordé à un principe durant une étape des études Delphi
- NR: Commentaires dénonçant le non respect d'un critère par un principe
- ?? : Commentaires que nous n'avons pas compris

Dans le cadre de notre recherche, nous ne retiendrons que les commentaires exprimant une opinion sur les principes car ces derniers nous permettront de comprendre le rôle de la mesure en génie logiciel. Les autres commentaires seront pris en considération dans des recherches subséquentes, lesquelles ne feront pas partie de cette activité de synthèse.

Parmi les commentaires retenus, nous éliminerons ceux ne se rapportant pas à la mesure. Pour ceux qui se rapportent à la mesure, nous les classifierons selon le type de mesure, tel que défini dans notre cadre conceptuel. Les catégories seront donc les suivantes (voir p. 12):

- la mesure pour la compréhension
- la mesure pour l'expérimentation
- la mesure pour le contrôle de projet
- la mesure pour l'amélioration du processus
- la mesure pour l'amélioration du produit
- la mesure pour les prédictions
- la mesure pour l'évaluation

Les commentaires pouvant être classés selon les catégories précédentes concernent le pourquoi de la mesure en génie logiciel. Cependant, certains commentaires ne pourront être classés selon ces catégories car ils concerneront la mesure d'un point de vue général ou encore le quoi et le comment mesurer. Ils se rapporteront probablement également à la question de l'utilité même de la mesure en génie logiciel. Nous nous intéresserons aussi à eux puisqu'ils nous permettront d'avoir une section du questionnaire portant sur la nécessité de mesurer en génie logiciel. Cette section sera importante car, avant de savoir quoi mesurer, il faut savoir s'il est nécessaire de mesurer.

Nous effectuerons ensuite une synthèse de ces données. Le but de cette synthèse sera d'identifier les incertitudes de la mesure en génie logiciel que les participants auront mentionnés dans les commentaires. Lors de cette synthèse, tous les commentaires retenus à l'étape précédente seront analysés pour y chercher les leçons, critiques, ou autres, qui y sont contenues. Nous identifierons les commentaires pouvant être regroupés. Nous identifierons également les nuances entre ces commentaires. Puis, nous analyserons ces nuances afin de déterminer si celles-ci doivent générer de nouvelles questions ou, sinon, si une seule question permettrait de couvrir toutes les nuances. La dernière étape de la synthèse consistera à décider des sujets sur lesquels porteront les questions. Le résultat final de cette synthèse sera présenté sous forme de matrices, dans les cas où il sera possible de réduire l'information.

Une fois la synthèse de données terminée, nous pourrons passer à l'étape de formulation des questions. La figure 3.3 illustre les étapes menant à la production du questionnaire. La soumission de ce questionnaire à certaines personnes constitue une recherche subséquente, ne faisant pas partie de notre projet, mais qui pourrait servir à éclaircir l'utilisation de la mesure en génie logiciel. Pour des résultats plus viables, deux autres chercheurs rattachés au projet effectueront également une vérification de l'analyse des données qualitatives.

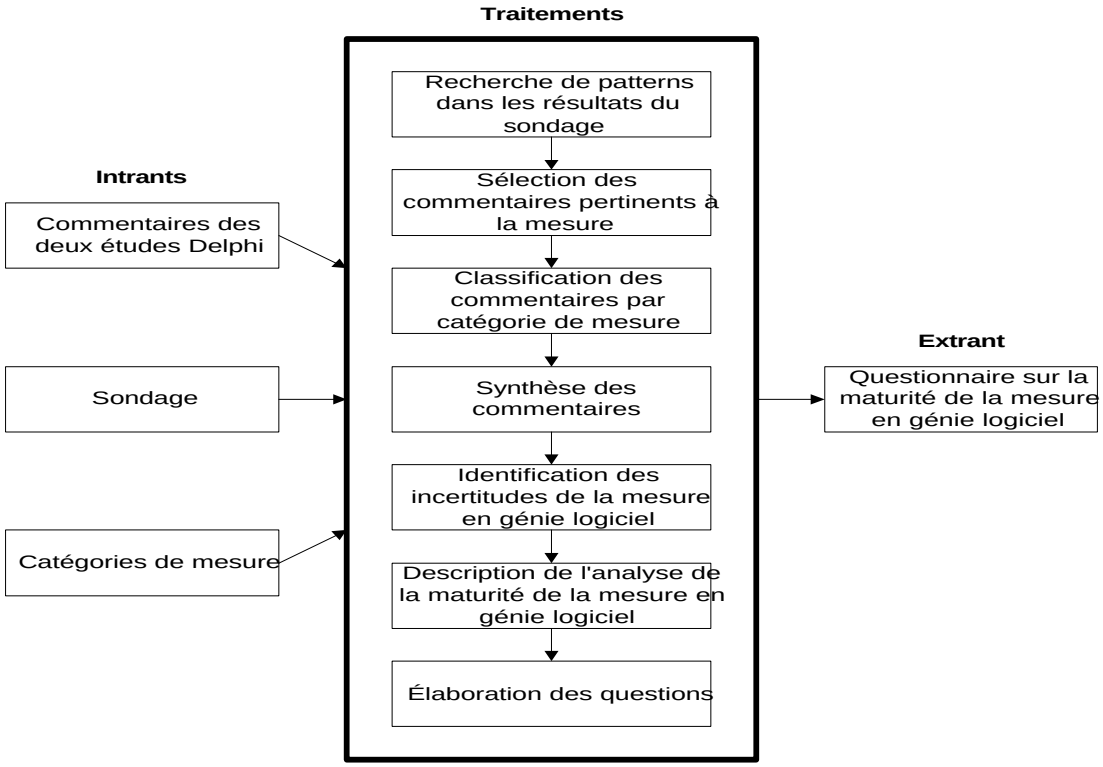


Figure 3.3 Étapes de production du questionnaire sur la mesure en génie logiciel

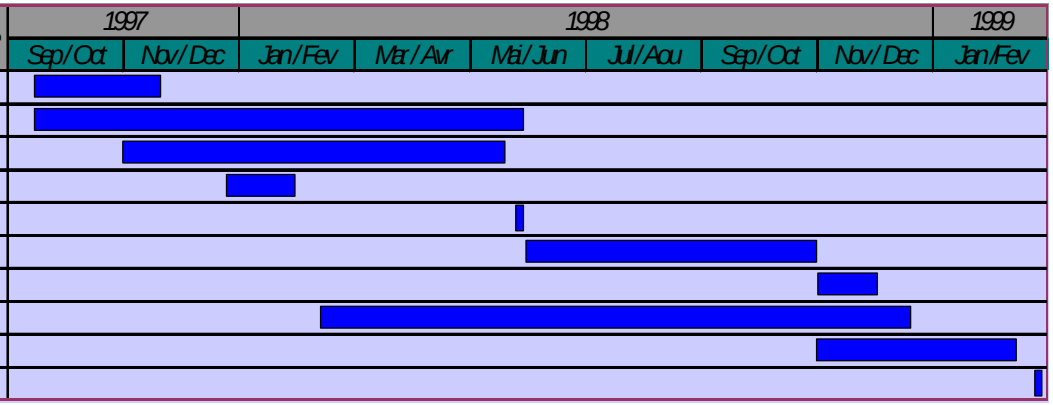
4. ÉCHÉANCIER

Au tableau 4.1, nous présentons l'échéancier du projet dans lequel sont représentées les étapes majeures du projet de recherche.

Tableau 4.1

Échéancier de l'activité de synthèse de Sibylle Wolff

ID	Nom de la tâche	Date Début	Date Fin	Durée
1	Choix et définition du sujet	15/09/97	20/11/97	49j
2	Recherche de documents pertinents	15/09/97	29/05/98	185j
3	Élaboration de la proposition	01/11/97	19/05/98	142j
4	Réalisation de l'étude Delphi 2	20/11/97	30/01/98	52j
5	Présentation de la proposition	29/05/98	29/05/98	1j
6	Modification de la proposition	01/06/98	30/10/98	110j
7	Sondage	01/11/98	01/12/98	22j
8	Analyse des données recueillies	15/02/98	15/12/98	217j
9	Rédaction du rapport final	01/11/98	11/02/99	75j
10	Présentation du rapport final	26/02/98	26/02/99	1j



CONCLUSION

Bien qu'une discipline telle le génie logiciel ait un rôle important à jouer dans l'amélioration des méthodes de développement de logiciels, cette branche reste encore jeune et mal définie. Nous voulons faire une analyse de la place de la mesure au sein de cette discipline. Nous pensons qu'une telle analyse permettrait d'identifier plus clairement certaines lacunes du génie logiciel. Cependant, notre travail se limitera à produire un questionnaire permettant d'y analyser la place de la mesure. Pour ce faire, nous utiliserons les principes fondamentaux du génie logiciel. La production des questions sera faite en se basant sur les différentes raisons de mesurer identifiées dans notre cadre conceptuel (voir la section intitulée POURQUOI MESURER EN GÉNIE LOGICIEL à la page 12 de ce document) ainsi que sur les commentaires effectués par les participants aux deux études Delphi et au sondage.

Notre cueillette de données se fait par le biais de deux études Delphi et d'un sondage qui nous donne accès aux opinions des experts en génie logiciel. Nous effectuerons l'analyse de données à la fois quantitatives et qualitatives.

Pour effectuer l'évaluation de la place de la mesure en génie logiciel, nous utiliserons une grille d'analyse basée sur les différentes raisons de mesurer identifiées dans ce document. Notre revue de littérature ainsi que des commentaires récoltés au cours des deux études Delphi nous permettront de remplir cette grille.

Certaines étapes du projet ont déjà été réalisées ou sont présentement en cours. Nous prévoyons de terminer notre recherche au mois d'août 1998.

Notre travail pourrait donner naissance à plusieurs autres projets de recherche sur les thèmes de la mesure en génie logiciel ou encore des principes fondamentaux du génie logiciel.

APPENDICE A

MESSAGE FINAL DE LA PREMIÈRE ÉTUDE DELPHI

----------Cover Letter----------
Greetings,

We would like to thank you for accepting the challenge of taking part in this Delphi study. As you can well understand, the quality of the results of such a study depends greatly on the input of the participants and your contribution has shown to be very valuable and thought provoking. You will find below the final results of this study including the list of participants, their biographical notices and a compilation of their comments from all three rounds.

The objective of this initiative is to identify a first set of Fundamental Principles in order to be able to better articulate or at least better organize the now somewhat confusing and overlapping standards of software engineering. Hence, results of this study will be presented and discussed at a workshop being held this week at the IEEE Third International Symposium and Forum on Software Engineering Standards (ISESS'97) in WalnutCreek, CA. The results of this workshop will also be tabled to an ISO/IECSC7 strategy meeting being held on completion of ISESS'97 to set future direction for the software engineering standardization effort within SC7,as well as to the IEEE Computer Society.

Thank you for your cooperation and best regards,

John Harauz,
General Chair, SES '96 Forum on Software Engineering Standards Issues
General Chair, ISESS '97 International Symposium on Software Engineering Standards
Vice-Chair, IEEE Software Engineering Standards Executive Committee

Alain Abran, Ph.D.
Director
Software Engineering Management Research Laboratory
Université du Québec à Montréal

----------Delphi Study FinalResults----------

Greetings,

With your cooperation, this Delphi study is now completed and this message presents the final results. A total of 14 recognized software engineering experts have taken part in this study. A compilation of their biographical information from the Web can be found in Appendix A. These experts in alphabetical order are:

- Motei Azuma, Waseda University, Japan
- Frederick P. Brooks, University of North Carolina at Chapel Hill,USA

- Robert N. Charette, ITABHI Corporation, USA
- Peter DeGrace, Author, USA
- Carlo Ghezzi, Politecnico di Milano, Italy
- Tom Gilb, Result Planning Limited, Norway
- Bev Littlewood, Center for Software Reliability, City University,London, UK.
- Steve MacDonell, University of Otago, New Zealand
- Tomoo Matsubara, Matsubara Consulting, Japan
- John Musa, Independent Consultant, USA
- Roger S. Pressman, R.S. Pressman & Associates, USA
- Mary Shaw, Carnegie-Mellon University, USA
- 1 participant has not responded to Round 3 due to other obligations
- 1 participant has chosen to remain anonymous

Please note that due to other obligations or the inaccessibility of the Internet, the number of participants per round is not 14. The number of participants per round is therefore:
- 13 participants in Round 1
- 10 participants in Round 2
- 12 participants in Round 3

The first list below contains the candidate Fundamental Principles on which the group has come to an consensus. The second group lists the candidate Fundamental Principles on which the group has not reached a consensus. In our opinion, both groups are equally valuable and will stimulate greatly our discussions this week in Walnut Creek, CA. We have arbitrarily determined that at least 10 out of the 12 Round 3 participants must have concurred with the rating for a consensus to be reached. We also encourage you to read Appendix B which contains a compilation of the participants's comments from all three rounds.

The 8 candidate Fundamental Principles of Software Engineering (in decreasing order of rating) for which a consensus has been reached are:

L. Since change is inherent to software, plan for it and manage it.
MEAN SCORE : 9,1
NUMBER OF YES VOTES (On 12) : 12

G. Invest in the understanding of the problem.
MEAN SCORE : 8,7
YES VOTES : 10

M. Since tradeoffs are inherent to software engineering, make them explicit and document them.
MEAN SCORE : 8,4
YES VOTES : 11

P. Uncertainty is unavoidable in software engineering. Identify and manage it.
MEAN SCORE : 8,0
YES VOTES : 11

K. Set quality objectives for each deliverable product.
MEAN SCORE : 7,7
YES VOTES : 11

E. Establish a software process that provides flexibility.
MEAN SCORE : 7,6
YES VOTES : 10

I. Minimize software components interaction.
MEAN SCORE : 7,3
YES VOTES : 11

O. The tools, methods, and support systems must be designed and selected to support the software engineers.
MEAN SCORE : 4,2
YES VOTES : 10

The 8 candidate Fundamental Principles of Software Engineering (in decreasing order of rating) for which a consensus has not been reached are:

B. Build with and for reuse.
MEAN SCORE : 8,0
YES VOTES : 9

H. Manage quality throughout the life cycle as formally as possible.
MEAN SCORE : 7,8
YES VOTES : 9

J. Produce software in a step-wise fashion.
MEAN SCORE : 7,7
YES VOTES : 8

A. Apply and use quantitative measurements in decision-making.
MEAN SCORE : 7,6
YES VOTES : 9

F. Implement a disciplined process and improve it continuously.
MEAN SCORE : 6,9
YES VOTES : 7

D. Define software artifacts rigorously.
MEAN SCORE : 6,4
YES VOTES : 8

C. Deal with different individual aspects of the problem by concentrating on each one separately.
MEAN SCORE : 4,9
YES VOTES : 6

N. The requirements must be firm and fixed.
MEAN SCORE : 3,3
YES VOTES : 9

We would like to thank you for taking the time and effort out of your busy schedule to take part in this Delphi study. Your contributions have all shown to be very valuable and has provided excellent input to our workshop in Walnut Creek, CA. Please be assured that we will keep you abreast of our future work on this topic. Please also feel free to contact us if you have any questions or comments or if you wish to pursue this topic with us.

Once again thank you very much and best regards.

Robert Dupuis
Directeur
Téléphone: (514) 987-3000 poste 3479
Télécopieur: (514) 987-8477
Maîtrise en informatique de gestion
Maîtrise en Génie logiciel
Université du Québec à Montréal

Pierre Bourque
Directeur adjoint
Laboratoire de recherche en gestion des logiciels
Departement d'informatique
Université du Québec à Montréal
Case postale 8888, succursale Centre-Ville
Montreal (Quebec) Canada
H3C 3P8
Telephone: (514) 987-3000 poste 0315
Telecopieur: (514) 987-8477
mailto: bourque.pierre@uqam.ca
http://www.info.uqam.ca/Labo_Recherche/lrgl.html

APPENDICE B

MESSAGE FINAL DE LA DEUXIÈME ÉTUDE DELPHI

Greetings,

We would like to thank you for accepting the challenge of taking part in this Delphi study. As you can well understand, the quality of the results of such a study depends greatly on the input of the participants, and your contribution has been shown to be very valuable and thought-provoking. You will find below the final results of this study, including the list of participants, their affiliation and their role as IEEE software engineering officials.

The objective of this initiative was to refine the results of the first Delphi study on Fundamental Principles of Software Engineering. We think that your numerous contributions will definitely make it possible to achieve our goal.

Thank you for your cooperation and best regards,

Leonard Tripp
Boeing Commercial Airplane Company
Chair, Software Engineering Standards Committee
1999 President
IEEE Computer Society

James W. Moore
The MITRE Corporation
Management Board, Software Engineering Standards Committee

---------- Delphi Study Final Results ----------

Greetings,

With your cooperation, this Delphi study is now completed and this message presents the final results. A total of 31 “IEEE software engineering officials” have taken part in this study. You will find below the list of participants, as well as their affiliation and their role within the IEEE Computer Society. We have done our best to ensure that these affiliations are current. Please notify us if this information is incomplete or out of date. These officials, in alphabetical order, are:

Maarten Boasson,
University of Amsterdam; department of Computer Science;
Hollandse Signaalapparaten BV;
IEEE Software Associate Editor.

Shawn Bohner,
Meta Group Inc.;
Technical Council on Software Engineering, Vice Chair Conferences.

Terry Bollinger,
The Mitre Corporation
IEEE Software Associate Editor.

Andy Bytheway,
Information Systems Research Center; Cranfield School of Management;
IEEE Software Associate Editor.

Carl Chang,
University of Illinois;
Member at-Large of the TCSE Executive Committee;
IEEE Software Editor-in-Chief, Emeritus.

James Cross II,
Auburn University; Computer Science & Engineering;
TCSE Secretary.

Peter Eirich,
Eirich Consulting;
TCSE Vice Chair, Standards.

Bill Everett,
Software Process & Reliability Engineering;
TCSE Committee Chair – Reliability Engineering.

Gene F. Hoffnagle,
IBM Corporation; Editor, IBM Systems Journal;
TCSE Chair.

Mehdi Jazayeri,
Technical University of Vienna;
Member of the editorial board of IEEE Transactions on Software Engineering.

Richard Kemmerer,
University of California; Dept. of Computer Science;
IEEE Transactions on Software Engineering Editor-in-Chief.

Barbara Kitchenham,
Dept. of Computer Science; University of Keele;
IEEE Software Associate Editor.

Reino Kurki-Suonio,
Software Systems Laboratory;
Member of the editorial board of IEEE Transactions on Software Engineering.

David John Leciston,
US Army Communications-Electronics Command;
TCSE Newsletter Editor.

Keith Marzullo,
Department of Computer Science & Engineering;
Member of the editorial board of IEEE Transactions on Software Engineering.

Nancy Mead,
Software Engineering Institute; Carnegie-Mellon University;
IEEE Software Associate Editor.

Stephen J. Mellor,
Project Technology Inc.;
IEEE Software Associate Editor.

Ware Myers,
IEEE Software Contributing Editor.

Michael Olsem,
USAF Software Technology Support Center;
TCSE Newsletter Editor.

Linda Ott,
Michigan Tech. University;
TCSE Newsletter Editor

Shari Lawrence Pfleeger,
Systems/Software, Inc.;
Member at-Large of the TCSE Executive Committee;
IEEE Software Contributing Editor;
Member of the editorial board of IEEE Transactions on Software Engineering.

Vaclav Rajlich,
Wayne State University; Dept. of Computer Science;
TCSE Executive Vice Chair.

Rami R. Razouk,
The Aerospace Corporation;
Member of the editorial board of IEEE Transactions on Software Engineering.

Sam Redwine,
Independent Consultant
TCSE Newsletter Editor.

Mary Lou Soffa,
Dept. of Computer Science; University of Pittsburgh;

Member of the editorial board of IEEE Transactions on Software Engineering.

David S. Wile,
USC/Information Sciences Inst.,
Member of the editorial board of IEEE Transactions on Software Engineering.

Linda Wills,
Georgia Institute of Technology; School of Electrical and Computer Engineering;
TCSE Committee Chair - Reverse Engineering

James Withey,
Software Engineering Institute; Carnegie Mellon University;
TCSE Committee Chair - Technology Transfer.

One participant chose to remain anonymous, one withdrew and one did not answer to Round 2.

The number of participants for each round of the study is:
30 participants in Round 1
29 participants in Round 2 (one person withdrew and one person was added, one person did not respond)

The first list below contains the candidate Fundamental Principles on which the group has come to a consensus on the median rating. The second group lists the candidate Fundamental Principles on which the group has not reached a consensus. In our opinion, both groups are equally valuable and will stimulate greatly our discussions in future projects. We have arbitrarily determined that at least 24 out of the 29 Round 2 participants must concur with the rating for a consensus to be reached. Appendix A contains a compilation of the participant's comments from the second round.

The 6 candidate Fundamental Principles of Software Engineering (in decreasing order of number of yes votes) for which a consensus has been reached on the median rating are:

G. INVEST IN THE UNDERSTANDING OF THE PROBLEM
Median rating from Round 1: 10
Yes Votes: 29

L. SINCE CHANGE IS INHERENT TO SOFTWARE, PLAN FOR IT AND MANAGE IT
Median rating from Round 1: 10
Yes Votes: 26

O. UNCERTAINTY IS UNAVOIDABLE IN SOFTWARE ENGINEERING. IDENTIFY AND MANAGE IT.
Median rating from Round 1: 10
Yes Votes: 25

I. MINIMIZE SOFTWARE COMPONENTS INTERACTION
Median rating from Round 1: 9
Yes Votes: 25

M. SINCE TRADEOFFS ARE INHERENT TO SOFTWARE ENGINEERING, MAKE THEM EXPLICIT AND DOCUMENT THEM
Median rating from Round 1: 9
Yes Votes: 25

N. TO IMPROVE DESIGN, STUDY PREVIOUS SOLUTIONS TO SIMILAR PROBLEMS
Median rating from Round 1: 9
Yes Votes: 24

The 9 candidate Fundamental Principles of Software Engineering (in decreasing order of number of yes votes) for which a consensus has not been reached on the median rating are:

C. CONTROL COMPLEXITY WITH MULTIPLE PERSPECTIVES AND MULTIPLE LEVELS OF ABSTRACTION
Median rating from Round 1: 8
Yes Votes: 23

J. PRODUCE SOFTWARE IN A STEPWISE FASHION
Median rating from Round 1: 8
Yes Votes: 23

D. DEFINE SOFTWARE ARTIFACTS RIGOROUSLY
Median rating from Round 1: 8
Yes Votes: 22

E. ESTABLISH A SOFTWARE PROCESS THAT PROVIDES FLEXIBILITY
Median rating from Round 1: 8
Yes Votes: 21

H. MANAGE QUALITY THROUGHOUT THE LIFE CYCLE AS FORMALLY AS POSSIBLE
Median rating from Round 1: 9
Yes Votes: 20

K. SET QUALITY OBJECTIVES FOR EACH DELIVERABLE PRODUCT
Median rating from Round 1: 8
Yes Votes: 20

F. IMPLEMENT A DISCIPLINED APPROACH AND IMPROVE IT CONTINUOUSLY
Median rating from Round 1: 8
Yes Votes: 19

B. BUILD WITH AND FOR REUSE
Median rating from Round 1: 9
Yes Votes: 17

A. APPLY AND USE QUANTITATIVE MEASUREMENTS IN DECISION-MAKING
Median rating from Round 1: 7
Yes Votes: 13

We would like to thank you for taking the time out of your busy schedule and making the effort to take part in this Delphi study. Your contributions have all been shown to be very valuable. Please be assured that we will keep you abreast of our future work on this topic. Also, please feel free to contact us if you have any questions or comments or if you wish to pursue this topic with us.

Once again, thank you very much and best regards.

Robert Dupuis, Ph.D.
Director
Master's degree Program in Software Engineering
Département d'informatique
Université du Québec à Montréal.
C.P. 8888, succursale Centre-Ville
Montréal (Québec) Canada
H3C 3P8
Téléphone: (514) 987-3000 ext. 3479
Fax: (514) 987-8477
mailto: dupuis.robert@uqam.ca

Pierre Bourque
Assistant Director
Software Engineering Management Research Laboratory
Département d'informatique
Université du Québec à Montréal
C.P. 8888, succursale Centre-Ville
Montréal (Québec) Canada
H3C 3P8
Téléphone: (514) 987-3000 ext. 0315
Fax: (514) 987-8477
mailto: bourque.pierre@uqam.ca
http://www.info.uqam.ca/Labo_Recherche/lrgl.html

Sibylle Wolff
Graduate student
Software Engineering Management Research Laboratory
Département d'informatique
Université du Québec à Montréal
CP 8888, succursale Centre-Ville
Montréal (Québec) Canada
H3C 3P8
mailto: m312455@er.uqam.ca

APPENDICE C

QUESTIONNAIRE DU SONDAGE

Greetings,

As 1999 President of the IEEE Computer Society, I am inviting you to complete the following survey on the topic of fundamental principles of software engineering. I believe this is important because it is hoped that the identification of a set of fundamental principles will promote the recognition of our profession as a well established engineering discipline. It will also provide a broader and richer framework for establishing relationships among groups of software engineering practice standards.

It is generally accepted that practice standards should be based on the observation, recording and consensual validation of implemented "best practices." This strategy, however, has resulted in the development of a body of software engineering standards that are sometimes said to be isolated and unconnected, because each standard often performs a local optimization of a single observed practice.

The Software Engineering Management Research Laboratory (SEMRL) of the Universite du Quebec a Montreal, on behalf of the IEEE, is currently doing research on this topic. A workshop was held to establish what a fundamental principle is and which criteria it should conform to. Then a Delphi (or consensus-based Internet) study was conducted among 14 renowned personalities of the Software Engineering community, to identify the principles of software engineering. A second workshop was held to eliminate or reformulate some of the principles and the criteria. Finally, a second Delphi (or consensus-based Internet) study was conducted among 31 IEEE Software Engineering officials in order to improve the principles (For more information, please see http://www.info.uqam.ca/Labo_Recherche/Lrgl/fpse/). From these studies, a list of fifteen candidate fundamental principles of software engineering has been compiled.

The results of the survey will help verify the relevance of these principles for practitioners and help determine which of these fifteen candidate principles are indeed fundamental. Results of this study will be submitted to the Software Engineering Standards Committee.

Thank you for your cooperation.

Leonard Tripp
Boeing Commercial Airplane Company
1999 President
IEEE Computer Society

SURVEY ON FUNDAMENTAL PRINCIPLES OF SOFTWARE ENGINEERING

The following questions will help us gather your opinions on the list of candidate fundamental principles. Please note that these principles are not accompanied by any explanatory sentences because this would make the survey too long. We are planning to hold a workshop in order to write these explanatory sentences. Some of the questions are concerned with your practical experience in software engineering. The survey will take approximately 20 minutes to complete. The IEEE Computer Society will maintain the anonymity of all respondents, and no emails or names will be transmitted to us.

What do we mean by software engineering?
We use the current definition as promoted by the IEEE Computer Society: It is "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e. the application of engineering to software."

What do we mean by "fundamental principle"?
A fundamental principle is less specific and more enduring than methodologies and techniques. It should be phrased to withstand the test of time. It should not contradict a more general engineering principle and should have some correspondence with "best practice." It should be precise enough to be capable of support and contradiction and should not conceal a tradeoff. It should also relate to one or more computer science or engineering concepts.

If you have any questions regarding this survey, please contact Sibylle Wolff (m312455@er.uqam.ca)

1. Number of years involved in Software Engineering: _____

2. Practical experience in the field of Software Engineering (number of years): _____

3. My education is in (you may choose more than one answer):
A. Engineering
B. Software Engineering
C. Computer Science
D. Information Systems
E. Other (specify) _____

4. Highest school/education degree:
A. High school
B. Bachelor's degree
C. Master's degree
D. Doctorate
E. Other (specify) _____

5. Country in which you live:_____

6. Primary line of business
A. Computers
B. Software
C. Communications systems and equipment
D. Aircraft, missiles, space and ground support equipment

E. Independent and university research, test and design laboratories and consultants (not connected with a manufacturing company)
F. Government agencies and armed forces
G. Companies using and/or incorporating any electronic products in their manufacturing, processing, research or development activities
H. Telecommunications services, telephone (including cellular)
I. Other (specify) _____

7. Job Function
A. Engineering Management
B. Project Leader, Manager, Engineer
C. Research & Development Manager
D. Research & Development Engineer
E. Software Design / Development
F. Computer Science
G. Education / Teaching
H. Design and Development Engineering
I. Other (specify) _____

8. Job Responsibility
A. Engineer or Scientific Management
B. Software: Science / Management /Engineering
C. Education / Teaching
D. Consulting
E. Other (specify) _____

9. Number of IS employees in your organization (your location):
_____

10. Type of software with which you are mostly involved:
A. MIS
B. Real-time system
C. Scientific
D. Other (specify) _____

11. To what extent is software engineering used in your organization:
A. Barely
B. A little
C. A fair amount
D. A lot
Extensively

We are asking you to rate each of the 15 candidate Fundamental Principles found below. Please assign a rating between 1 and 10 to each of the Principles (10 meaning that you TOTALLY AGREE that this is a Fundamental Software Engineering Principle and 1 meaning that you TOTALLY DISAGREE that this is a Fundamental Software Engineering Principle). These principles are listed in alphabetical order.

A. APPLY AND USE QUANTITATIVE MEASUREMENTS IN DECISION-MAKING

Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____
B. BUILD WITH AND FOR REUSE
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

C. CONTROL COMPLEXITY WITH MULTIPLE PERSPECTIVES AND MULTIPLE LEVELS OF ABSTRACTION
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

D. DEFINE SOFTWARE ARTIFACTS RIGOROUSLY
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

E. ESTABLISH A SOFTWARE PROCESS THAT PROVIDES FLEXIBILITY
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

F. IMPLEMENT A DISCIPLINED APPROACH AND IMPROVE IT CONTINUOUSLY
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

G. INVEST IN THE UNDERSTANDING OF THE PROBLEM
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

H. MANAGE QUALITY THROUGHOUT THE LIFE CYCLE AS FORMALLY AS POSSIBLE
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

I. MINIMIZE SOFTWARE COMPONENTS INTERACTION
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

J. PRODUCE SOFTWARE IN A STEPWISE FASHION
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

K. SET QUALITY OBJECTIVES FOR EACH DELIVERABLE PRODUCT
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

L. SINCE CHANGE IS INHERENT TO SOFTWARE, PLAN FOR IT AND MANAGE IT
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

M. SINCE TRADEOFFS ARE INHERENT TO SOFTWARE ENGINEERING, MAKE THEM EXPLICIT AND DOCUMENT THEM
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

N. TO IMPROVE DESIGN, STUDY PREVIOUS SOLUTIONS TO SIMILAR PROBLEMS
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

O. UNCERTAINTY IS UNAVOIDABLE IN SOFTWARE ENGINEERING. IDENTIFY AND MANAGE IT
Score (1means you TOTALLY DISAGREE and 10 means you TOTALLY AGREE): _____

Additional comments (optional):

_____

_____

APPENDICE D

CADRE DE BASILI MODIFIÉ

Tableau E.1

Cadre de Basili pour l'activité de synthèse de Sibylle Wolff

<table>
<tr><th colspan="6">Définition</th></tr>
<tr><th>Motivation</th><th>Objet</th><th colspan="2">Objectif</th><th colspan="2">Utilisateurs de la recherche</th></tr>
<tr><td>Mieux définir la discipline du génie logiciel</td><td>La mesure en génie logiciel</td><td colspan="2">• Produire un questionnaire sur la place de la mesure en génie logiciel
• Identifier les incertitudes concernant la mesure en génie logiciel</td><td colspan="2">• Praticiens en génie logiciel
• Concepteurs de curriculum en génie logiciel
• Organismes d'accréditation
• LRGL dans le cadre du projet SWEBOK
• Chercheurs sur la mesure en génie logiciel</td></tr>
<tr><th colspan="6">Planification</th></tr>
<tr><th colspan="2">Étapes du projet</th><th colspan="2">Intrants</th><th colspan="2">Livrables</th></tr>
<tr><td colspan="2">Deuxième étude Delphi</td><td colspan="2">• Première étude Delphi
• experts des comités IEEE en génie logiciel</td><td colspan="2">Analyse des données recueillies (scores et commentaires) des participants</td></tr>
<tr><td colspan="2">Rédaction de la proposition</td><td colspan="2">Revue de littérature sur le génie, le génie logiciel, la mesure en génie logiciel, les principes fondamentaux du génie logiciel</td><td colspan="2">Proposition</td></tr>
<tr><td colspan="2">Sondage</td><td colspan="2">• praticiens du génie logiciel, membres des comités IEEE
• Delphi 1, Delphi 2</td><td colspan="2">Analyse des données recueillies (recherche de patterns, application de tests statistiques)</td></tr>
<tr><td colspan="2">Rédaction du rapport final</td><td colspan="2">• Revue de littérature sur le génie, le génie logiciel, la mesure en génie logiciel, les principes fondamentaux du génie logiciel
• Delphi 1, Delphi 2 et sondage</td><td colspan="2">Rapport final</td></tr>
<tr><th colspan="6">Exécution</th></tr>
<tr><th>Delphi 2</th><th>Revue de littérature</th><th>Sondage</th><th>Analyse des données recueillies</th><th>Description des incertitudes concernant la mesure en génie logiciel</th><th>Production du questionnaire sur la mesure en génie logiciel</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><th colspan="6">Interprétation</th></tr>
<tr><th colspan="2">Contexte d'interprétation</th><th colspan="2">Extrapolation des résultats</th><th colspan="2">Travaux subséquents</th></tr>
</table>

.		• Rédaction d'articles sur le sujet • Analyse des principes fondamentaux selon d'autres perspectives • Rédaction de textes explicatifs devant accompagner les principes fondamentaux retenus. • Soumission du questionnaire sur la mesure en génie logiciel à des personnes du domaine

RÉFÉRENCES

Bagert, Don. 1998. «Texas Board of Professional Engineers», *Forum for Advancing Software engineering Education (FASE)*, Vol. 8, no 4, (15 avril).

Boehm, Barry W. 1989. «Seven Basic Principles of Software Engineering», *Journal of Systems and Software*, Vol. 3, no 1, (mars), p. 3-24.

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

Brooks, Frederick P. 1995. *The Mythical Man-Month*, (anniversary edition) Addison-Wesley.

Coallier, F. et P. N. Robillard. 1985. «An Overview of Software Engineering Standards Activities in the IEEE», p. 134-138.

Cox, Brad J. 1990. «Planning the Software Industrial Revolution», *IEEE Software*, (novembre).

Dakin, Karl. 1997 «Software "Engineer"? Time Will Tell», *IEEE Software*, Vol. 14, no 13, (mai-juin), p. 105-106.

Davis, Allan M. 1995. *201 Principles of Software Development*, McGraw-Hill.

Davis, Duane. 1995. *Business Research for Decision Making*, (quatrième édition) Duxbury Press.

Delbeck, André L., A. H. Van de Ven et D. H. Gustafson. 1975. *Group Techniques for Program Planning*, Green Briar Press.

Dupuis, Robert, Pierre Bourque, Alain Abran et James W. Moore. 1997. «Principes fondamentaux du génie logiciel: Une étude Delphi»: Acte de la conférence *Le génie logiciel et ses applications, dixièmes journées internationales* (Paris, 3-5 décembre).

Emerson, Howard P. et Douglas C.E. Naehring. 1988. *Origins of Industrial Engineering: The Early Years of a Profession*, Industrial Engineering & Management Press.

Eventoff, William. 1996. «Software Engineering Standards: Needs and Priorities»: Texte préparé pour l'atelier sur les principes fondamentaux du génie logiciel durant le *Forum on Software Engineering Standards Issues* (Montréal, 21-25 octobre 1996).

Fenton, Norman E. et Shari Lawrence Pfleeger. 1997. *Software Metrics: A Rigorous & Practical Approach*, International Thomson Computer Press.

Glass, Robert L. 1996. «The Relationship Between Theory and Practice in Software Engineering», *Communications of the ACM*, Vol. 39, no 11, (novembre), p. 11-13.

Huberman, A. Michael et Matthew B. Miles. 1991. *Analyse des données qualitatives: Recueil de nouvelles méthodes*, De Boeck Université.

IEEE Std 610.12. 1990. *Standard glossary of software engineering methodology*.

ISO/IEC 9126. 1991. *Software product evaluation - Quality characteristics and guidelines for their use*.

Jabir. 1998. «A Search for Fundamental Principles of Sofware Engineering»: Rapport d'un atelier réalisé au *Forum on Software Engineering Standards Issues* (Montréal, 21-25 octobre 1996), publié dans *Computer Standards and Interfaces*, Vol. 19, no 2, p. 155-160 (Les participants à cet atelier se sont donnés le nom collectif de Jabir).

Jackson, M. 1998. «Will There Ever Be Software Engineering?», *IEEE Software*, Vol. 15, no 1, (janvier-février), p. 36-39.

Jacquet, Jean-Philippe et Alain Abran. 1997. «From Software Metrics to Software Measurement Methods: A Process Model», *Third International Symposium and Forum on Software Engineering Standards*, ISESS'97, Walnut Creek (CA), juin.

Jacquet, Jean-Philippe, Alain Abran et Robert Dupuis. 1997. «Une analyse structurée des méthodes de validation de métriques», Acte de la conférence *Le génie logiciel et ses applications, dixièmes journées internationales* (Paris, 3-5 décembre).

Kirby, Richard Shelton, Sidney Withington, Arthur Burr Darling et Frederick Gridley Kilgour. 1990. *Engineering in History*, Dover Publications Inc., New York.

Laframboise, Lucie. 1996. «Grille d'évaluation des facteurs de risque d'un programme de mesures en génie logiciel», Rapport d'activité de synthèse de la Maîtrise en informatique de gestion, Montréal, Université du Québec à Montréal, 122 p.

Magee, S. et L. Tripp. 1997. *Guide to Software Engineering Standards and Specifications*, Artech House.

McConnell, Steve. 1996. «Who Cares About Software Construction?», *IEEE Software*, Vol. 13, no 1, (janvier).

McConnell, Steve. 1996. «Daily Build and Smoke Test», *IEEE Software*, Vol. 13, no 4, (juillet).

McConnell, Steve. 1997. «Software's Ten Essentials», *IEEE Software*, (mars), p.143-144.

Moore, James W. 1996. «A "Principled" Approach to Software Engineering Standardization»: Texte préparé pour l'atelier sur les principes fondamentaux du génie logiciel durant le *Forum on Software Engineering Standards Issues* (Montréal, 21-25 octobre 1996).

Moore, James W. 1998. *Software Engineering Standards: A User's Road Map*, IEEE Computer Society Press.

Nadeau, Marc-André. 1982. «La technique Delphi: une technique utile», Département de mesure et évaluation, Université Laval, Vol. 1, no 5, (juin).

Oman, Paul et Shari Lawrence Pfleeger. 1997. *Applying Software Metrics*, IEEE Computer Society Press.

Parnas, David Lorge. 1997. «Software Engineering: An Unconsummated Marriage», *Communications of the ACM*, Vol. 40, no 9, (septembre), p. 128.

Pfleeger, Shari Lawrence, Norman Fenton et Stella Page. 1994. «Evaluating Software Engineering Standards», *IEEE Computer*, (septembre), p. 71-79.

Pressman, Roger S. 1997. *Software Engineering: A Practitioner's Approach*, (quatrième édition) McGraw-Hill.

Ramamoorthy, C. V., Wei-Tek Tsai. 1996. «Advances in Software Engineering», *IEEE Software*, (octobre), p. 47-58.

Read, Wallace S. et Jay Iorio. 1994. «Streamlining the Standards Development Process», *IEEE Canadian Review*, (printemps-été).

Robinson, Gary S., Carl Cargill. 1996. «History and Impact of Computer Standards», *IEEE Software*, (octobre), p. 79-85.

Schiemann, William A. et John H. Lingle. 1998. «Seven Greatest Myths of Measurement», *IEEE Engineering Management Review*, (printemps), p. 114-116.

Shaw, Mary. 1990. «Prospects for an engineering discipline of software», *IEEE Software*, p. 930-940.

Tripp, Leonard L. et Peter Voldner. 1995. «A Market-Driven Architecture For Software Engineering Standards», Proceedings of the *Second International Software Engineering Standards Symposium* (ISESS'95), Montréal, Québec, IEEE Computer Society Press.

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

Wasserman, Anthony I. 1996. «Toward a Discipline of Software Engineering», *IEEE Software*, Vol.13, no 6, (novembre), p. 23-31.