UNIVERSITÉ DU QUÉBEC À MONTRÉAL

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

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

PAR
SIBYLLE WOLFF

JUILLET 1999

Rapport d'activité de synthèse approuvé par

_____

Robert Dupuis

Directeur de la maîtrise en informatique de gestion

Université du Québec à Montréal

_____

Pierre Bourque

Directeur de la recherche appliquée

Laboratoire de recherche en gestion des logiciels

Université du Québec à Montréal

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui m'ont aidé à la réalisation de mon activité de synthèse dans le cadre de ma maîtrise en informatique de gestion.

Je tiens à remercier tout particulièrement :

Messieurs Robert Dupuis et Pierre Bourque, mes directeurs, pour leur support, leur disponibilité et tous les bons conseils qu'ils m'ont donnés durant mes travaux. Je tiens à remercier également monsieur Alain Abrain, professeur et directeur du Laboratoire de recherche en gestion des logiciels

Tous les membres du Laboratoire de recherche en gestion des logiciels, particulièrement Michèle Hébert, pour leur support et leur collaboration.

Messieurs Leonard Tripp, James Moore et John Keaton, de la IEEE Computer Society, sans qui les études sur les principes fondamentaux n'auraient pas été possibles.

Monsieur Bertrand Fournier pour ses conseils dans l'analyse statistique des résultats des différentes études.

Tous mes parents, amis et collègues qui m'ont toujours soutenu tout au long de ma recherche et à ceux et celles qui ont bien voulu lire mon texte et me permettre de l'améliorer.

Un merci tout particulier à ma mère et à Marie-France qui m'ont rappelé que l'amour donne des forces.

TABLE DES MATIÈRES

LISTE DES FIGURES ET DES TABLEAUX

AVANT-PROPOS

« *I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory king : it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of* ***science****, whatever the matter may be.* »

Lord Kelvin, inventeur de l'échelle de valeur absolue pour la mesure des températures.[1]

[1] Lord Kelvin, « Popular Lectures and Addresses », Volume 1, 1889.

RÉSUMÉ

Le génie logiciel est une discipline encore mal établie. Contrairement à ce qu'indique son nom, il n'est pas accepté comme branche du génie. En fait, est-ce du génie? Nous nous sommes posé la question et nous avons pensé qu'un élément de réponse était d'y analyser la place de la mesure, puisque cette dernière occupe généralement une place de taille en génie.

M. Parnas, une autorité du génie logiciel, avait suggéré de tenter de déterminer les principes fondamentaux du génie logiciel afin de faire évoluer le domaine en tant que profession reconnue. L'identification de ces principes permettrait ainsi de s'attaquer à des problèmes précis tel que le manque de cohérence dans le corpus de normes, la reconnaissance de la discipline comme faisant partie du génie, l'établissement des programmes de formation en génie logiciel et la définition des compétences d'un ingénieur en logiciel. Une étude Delphi a été effectuée par des chercheurs du laboratoire de recherche en gestion de logiciels de l'Université du Québec à Montréal afin d'arriver à identifier une première liste de principes. Notre recherche s'insère dans ces démarches. Puisque la mesure tient habituellement une place importante dans les disciplines du génie, nous voulons en évaluer la place au sein des principes fondamentaux du génie logiciel dans le but de mieux cerner le niveau d'évolution de la discipline. Les principes dont on parle ici sont ceux identifiés par l'étude Delphi mentionnée plus haut et par les deux études subséquentes dont nous parlons dans le paragraphe qui suit.

Notre objectif principal est de produire une série de questions sur des incertitudes concernant la mesure en génie logiciel. Ces questions pourront servir de base pour des recherches subséquentes dans le domaine. Afin d'atteindre cet objectif, nous avons entrepris certaines démarches : Nous avons utilisé les résultats de l'étude Delphi mentionnée plus haut afin de réaliser une deuxième étude Delphi et un sondage. Ceux-ci ont permis de consolider les résultats de la première étude Delphi. Nous avons également effectué une revue de littérature sur les thèmes suivants : le génie, le génie logiciel, les principes fondamentaux du génie logiciel, le génie logiciel en tant que branche du génie, la place de la mesure en génie logiciel.

Ces démarches nous ont permis de conclure que la mesure n'est pas aussi utilisée en génie logiciel que dans les autres branches du génie. De plus, lorsqu'elle est utilisée, ce n'est pas toujours fait de manière rigoureuse, selon les règles de l'art. Notre recherche nous a permis d'obtenir une liste riche et longue d'incertitudes concernant la mesure en génie logiciel. Ce travail pourra servir de point de départ pour d'autres recherches sur le sujet, bien qu'il n'ait pas la prétention d'identifier toutes les incertitudes de la mesure dans ce domaine. Par contre, il nous montre qu'il y a encore beaucoup à faire afin d'améliorer la discipline du génie logiciel.

ABSTRACT

Software engineering is still not well established. Contrarily to what its name states, it is not yet accepted as an engineering discipline. In fact, is it engineering? We have asked ourselves the question. We thought that a way to answer this question was to analyze the role of software measurement, since measurement usually plays an important role in engineering.

Mr Parnas, an authority in software engineering, had recommended to determine fundamental principles of software engineering in order to help software engineering evolve into a recognized discipline. Identification of the principles would help solve problems such as the lack of consistence in standards, the recognition of software engineering as an engineering discipline, the establishment of curricula in software engineering, as well as the definition of core competencies of a software engineer. A Delphi study was conducted by members of the Software Engineering Management Research Laboratory of the University of Quebec in Montreal in order to identify a first list of principles. Since measurement plays an important role in engineering, we evaluated its role in fundamental principles of software engineering, in order to better understand the level of evolution of the discipline. By fundamental principles we mean the principles that have been identified by the Delphi study mentioned earlier and by two subsequent studies described in the following paragraph.

Our main goal is to produce a list of questions on the uncertainties of measurement in software engineering. These questions will serve as a basis for future work on the subject. To achieve our goal, we took the following steps : We used the results of the Delphi study mentioned earlier to conduct a second Delphi study and a survey. Those two studies allowed to consolidate the results of the first Delphi study. We also conducted a literature review on the following subjects : engineering, software engineering, fundamental principles of software engineering, software engineering as an engineering discipline and the role of measurement in software engineering.

These studies have allowed used to conclude that measurement is not as extensively used in software engineering as in other disciplines of engineering. Furthermore, when it is used, it is not always done in a rigorous manner, following standard measurement practices. Our research has allowed us to obtain a rich and important list of uncertainties concerning measurement in software engineering. This work will be used as a basis for further research on the subject, even though it has no pretension of identifying all uncertainties of measurement in software engineering. However, it shows us that there is still a lot of work to do to improve the discipline of software engineering.

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 le «International Symposium on Software Engineering Standards» (ISESS'97) 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[2] et plus précisément auprès de membres impliqués dans le génie logiciel. Une partie de notre recherche a consisté à réaliser cette deuxième étude Delphi. Celle-ci a permis d'épurer les résultats obtenus lors de la première étude. Puis, nous avons tenté 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 nous sommes concentrée plutôt sur l'aspect mesure de cette discipline. Nous avons voulu identifier les principes fondamentaux du génie logiciel, entre autres, 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 général, nous avons orienté 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 effectuer une première tentative d'évaluation du degré de maturité de cette discipline.

Dans les prochaines sections, nous décrirons la recherche que nous avons effectuée. 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

---

[2] http:\\www.computer.org

ainsi que les démarches nous permettant d'atteindre nos objectifs. Puis nous présenterons une synthèse des données quantitatives recueillies ainsi que l'analyse de la place de la mesure au sein du génie logiciel. Une synthèse de la recherche permettra d'avoir une vue générale sur les résultats, l'apport, les limites de la recherche ainsi que l'apprentissage que nous en avons tiré. Nous terminerons ce document par 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, ainsi que l'étude de telles approches (IEEE 610.12, 1990). Cependant, 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 cette activité de synthèse, 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 quelles sont les compétences et connaissances d'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 avons entrepris 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 nous semble 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 mesures du logiciel proposées 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 mesures». 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 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 de 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.

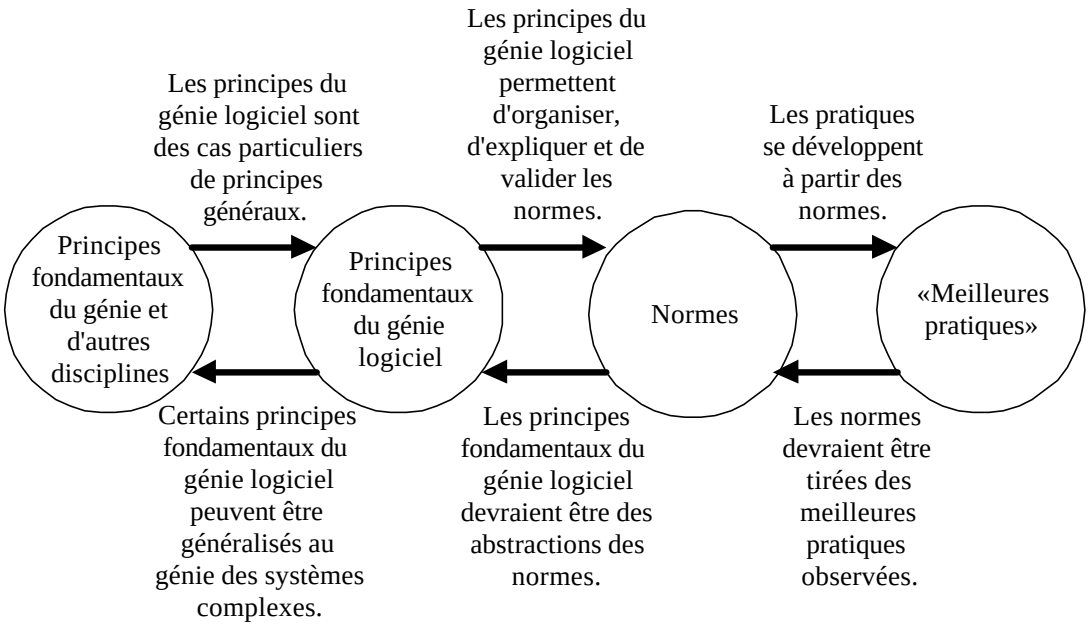

Figure 1.1 Place des principes fondamentaux (Tirée de Jabir, 1998)

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. Sur ce sujet, M. Parnas 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).

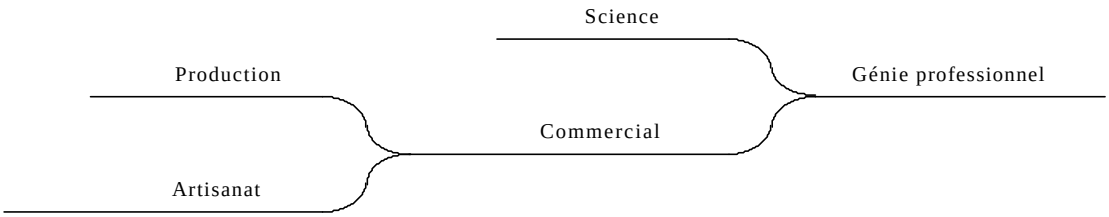

Figure 1.2 Évolution d'une discipline du génie (Tiré de Shaw, 1990)

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[3] 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 mesures 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.

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.

---

[3] http:\\www.lrgl.uqam.ca

2. OBJECTIFS

L'objectif principal de notre recherche était de produire un questionnaire sur la place de la mesure en génie logiciel. Ce questionnaire permettrait d'analyser la vision de personnes du domaine sur la place de la mesure à l'intérieur du génie logiciel. Nous nous sommes intéressée 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. Nous rappelons que ce questionnaire n'est pas quelque chose de prêt à être soumis à un ensemble de participants. Il constitue plutôt un point de départ pour des investigations plus profondes sur la place qu'occupe la mesure en génie logiciel.

Afin d'arriver à produire ce questionnaire, nous avons dû identifier les incertitudes concernant la mesure en génie logiciel. Ceci constituait un sous-objectif de notre recherche. Plusieurs aspects abordés sont le quoi, le quand, le comment (procédures et outils), le pourquoi (dans quels buts) de la mesure en génie logiciel

Pour atteindre ces objectifs, nous avons utilisé les résultats de recherches 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 ont permis d'identifier les incertitudes de la mesure en génie logiciel et, à partir de là, de produire un questionnaire. Nous avons alors analysé 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

3.1 CADRE CONCEPTUEL

Un 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 prévoyait 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 3.2), 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 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, par exemple 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 EFFECTUÉES AFIN D'ATTEINDRE LES OBJECTIFS

Dans cette section, nous abordons les différentes méthodes que nous avons utilisées pour notre recherche. Nous donnons d'abord une vue globale des démarches que nous avons entreprises afin d'atteindre les objectifs que nous nous sommes fixés. Rappelons que notre objectif principal est d'évaluer la place de la mesure en génie logiciel au sein des principes fondamentaux.

La cueillette de données provient d'une étude Delphi déjà réalisée, et également d'une seconde étude Delphi et d'un sondage que nous avons menés auprès des praticiens du génie logiciel.

Parallèlement à notre cueillette de données, nous avons procédé 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 avons consultés, voir la section 3.2.4 du présent document, portant sur la revue de la littérature.

Dans la dernière étape de notre recherche, nous avons utilisé les résultats des deux études et du sondage ainsi que des connaissances acquises dans la revue de littérature pour identifier les incertitudes de la mesure en génie logiciel. Les commentaires se rapportant à la mesure ont été classés d'après les différentes raisons de mesurer identifiées en génie logiciel (Oman et Pfleeger, 1997; Fenton et Pfleeger, 1997). Une description de l'analyse de la place de la mesure en génie logiciel a également été effectuée. À partir de cette analyse, nous avons produit un questionnaire pouvant permettre d'approfondir l'analyse de la place de la mesure au sein du génie logiciel. Les détails de l'analyse des données qualitatives se trouvent à la page 25 de ce document.

La figure 3.1 illustre les différentes étapes ayant précédées notre recherche. Cette figure s'inspire d'une figure tirée de l'article *Principes fondamentaux du génie logiciel : Une étude Delphi* (Dupuis *et al*., 1997). La figure 3.2 illustre les grandes étapes de cette activité de synthèse.

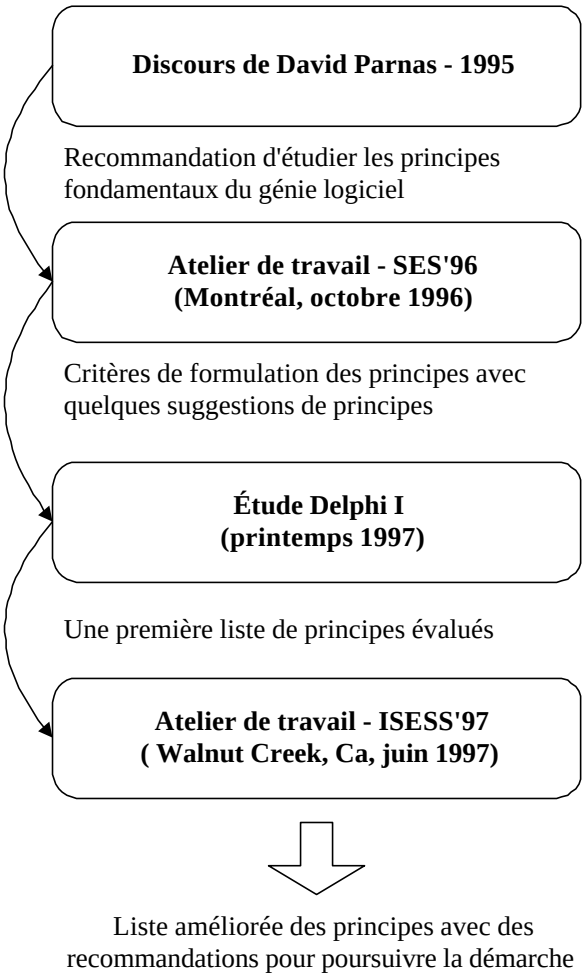

Figure 3.1 Travaux effectués sur l'identification des principes fondamentaux du génie logiciel (Étapes ayant précédées notre recherche) (Tirée de Dupuis *et al*., 1997)

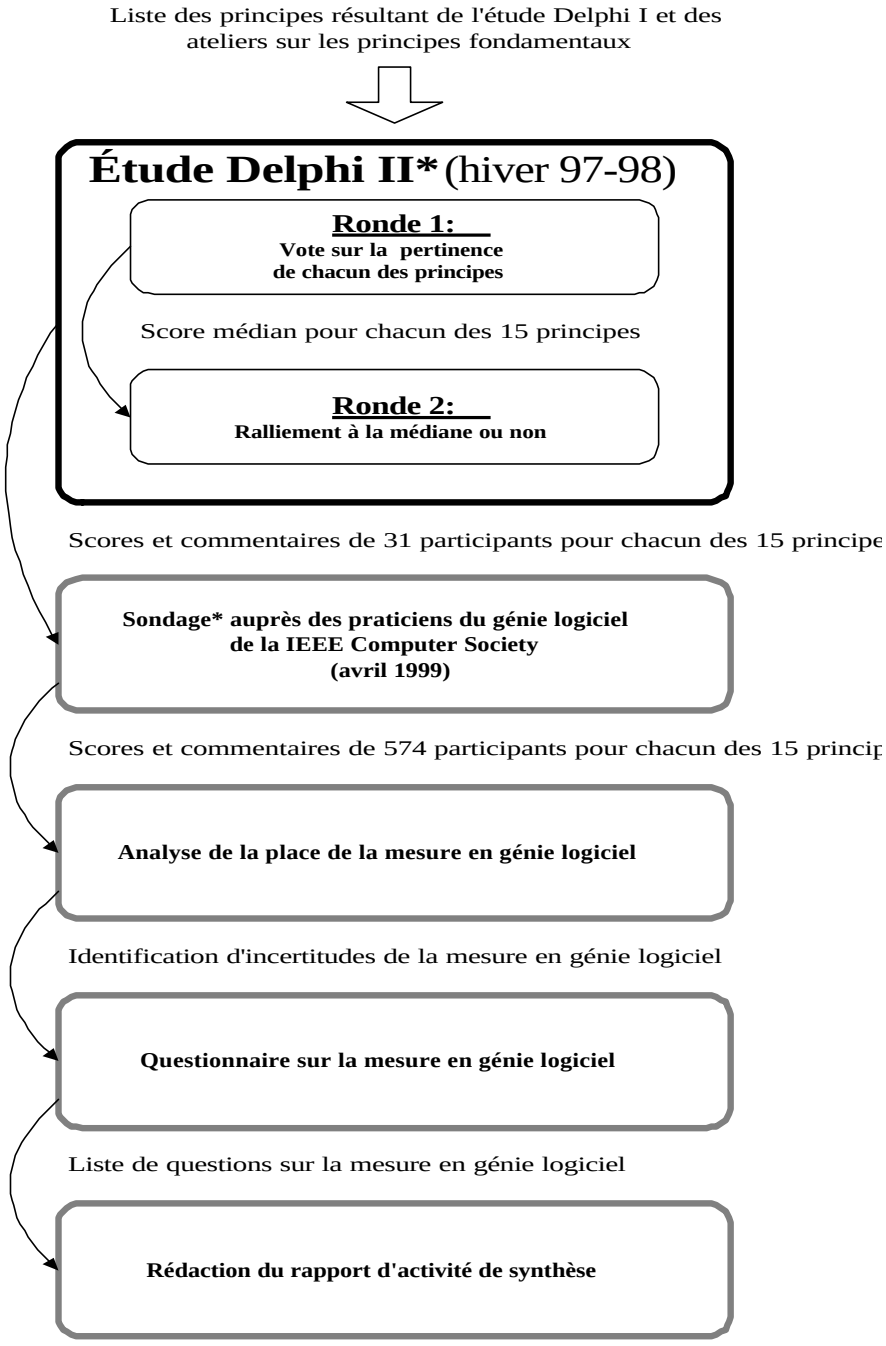

Figure 3.2 Grandes étapes de la recherche sur la place de la mesure au sein des principes fondamentaux du génie logiciel

---

* Étape réalisée en collaboration avec Pierre Bourque et Robert Dupuis

Nous avons commencé par donner une idée globale des démarches que nous avions entreprises afin d'atteindre nos objectifs. Dans la présente section, chacune des ces étapes sera approfondie dans le but de présenter en détail la démarche que nous avons suivie. Nous débuterons par une description des étapes ayant précédées notre recherche. Les autres étapes qui seront décrites sont celles de la cueillette de données qui se retrouvent à la figure 3.2 et qui ont été réalisées dans le cadre de cette activité de synthèse. Ce sont la deuxième étude Delphi et le sondage auprès des praticiens du génie logiciel. Nous donnerons également une idée de la revue de la littérature que nous avons effectuée. Nous ne décrirons pas l'étape de rédaction du rapport d'activité de synthèse car cela n'apporterait aucun élément nouveau à la démarche entreprise pour l'atteinte de nos objectifs.

3.2.1 Étapes précédant notre recherche

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 en ce domaine. 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 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, décrites dans les paragraphes suivants.

Lors d'un atelier de travail durant le «Forum on Software Engineering Standards Issues» de 1996 (SES'96)[4], 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).

---

[4] http://www.lrgl.uqam.ca/ses96.html

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 correspondait au complet désaccord d'un individu avec ce principe et 10 à son 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 se ralliaient à 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

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 l'influence 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.

Par la suite, un deuxième atelier s'est tenu au «International Symposium on Software Engineering Standards» (ISESS'97) de juin 1997[5]. 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. Dans le chapitre portant sur les résultats des données recueillies, nous vous présentons la liste des principes qui ont résulté des étapes décrites ci-dessus et que nous avons utilisé pour nos recherches subséquentes :

3.2.2 Réalisation de la deuxième étude Delphi

Comme le montre la figure 3.1, une première étude Delphi et un atelier de travail avaient été effectués afin d'identifier les principes fondamentaux du génie logiciel. De ces démarches était ressortie une liste de quinze principes. Lors de la présentation des résultats de la première étude Delphi à l'atelier de travail, les membres de cet 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 d'affiner 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 s'ils étaient des principes fondamentaux du génie logiciel ou pas, ou encore, s'il était nécessaire de les reformuler. Cette étude a été réalisée par l'auteur, en collaboration avec Pierre Bourque et Robert Dupuis.

Avant de débuter la deuxième étude Delphi, nous avons pris soin de la tester en la soumettant à certains professionnels en génie logiciel de notre entourage. Des personnes contactées, cinq ont accepté de nous fournir des commentaires nous permettant d'améliorer notre étude.

---

[5] http://nssc.llnl.gov/SESI/ses97/ses97.htm

Comme nous l'avons dit plus haut, l'échantillon de la deuxième étude Delphi devait être composé des membres des comités de la IEEE Computer Society qui sont fortement impliqués dans le génie logiciel. Nous avons donc sélectionné les membres des comités éditoriaux des journaux IEEE Software et IEEE Transactions on Software Engineering ainsi que les responsables des groupes d'intérêt du Technical Committee on Software Engineering. Nous avons obtenu une liste de 76 personnes répondant à ces critères. Nous leur avons envoyé une invitation à partir à cette étude Delphi. Sur les 76 personnes contactées, 31 personnes ont accepté de prendre part à cette étude. L'absence d'un certain nombre de personnes étant dû à de mauvaises adresses électroniques. D'autres nous ont répondu qu'elles n'avaient malheureusement pas le temps d'y participer. Quelques-unes, en désaccord avec l'étude ou avec la manière de la conduire n'ont pas voulu y participer. Enfin, d'autres ne nous ont pas répondu.

Contrairement à la première étude, la deuxième comprenait deux tours. Ces premier et deuxième tours correspondaient respectivement aux deuxième et troisième tours de la première étude. Lors du premier tour, une liste de principes avait été fournie aux participants ainsi qu'une liste de critères que devaient 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 dû noter, sur une échelle de 1 à 10, les différents principes proposés, selon l'importance qu'ils leur accordaient en génie logiciel. Les participants nous ont ensuite retourné leurs scores et commentaires. Puis, nous avons 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 se ralliaient à la médiane 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.

Nous avons dit plus haut que 31 personnes avaient participé à cette deuxième étude Delphi. Cependant, toutes non pas participé aux deux tours. En effet, 30 personnes ont pris part à la première ronde. Pour la deuxième ronde, une nouvelle personne y a pris part, une autre a choisi de se retirer et une dernière n'a pas répondu, ce qui donne un total de 29 participants pour cette ronde. Nous avions décidé qu'il y aurait consensus à partir de 24 réponses positives sur 29 personnes ayant pris part à la deuxième ronde.

Au deuxième tour, les principes ont été à nouveau soumis aux participants, mais cette fois dans le but de leur faire dire s'ils se ralliaient à la médiane 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.

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 dévoilée à tous les participants à la fin de l'étude en même temps que les résultats. Cependant, les commentaires sont restés impersonnels. Sur les 31 particpants, 28 ont accepté, 2 ont refusé et 1 s'est retiré. Vous pouvez voir la liste des noms à l'appendice B à la page 90 du présent document.

Les résultats des deux études Delphi consistent en une cote médiane ou médiane pour chacun des principes, le nombre de personnes acceptant cette cote comme respectant l'importance du principe en 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 et également de discuter de la place de la mesure en génie logiciel. Ce travail d'affinement 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 Par ailleurs, ce sont en grande partie ces commentaires que nous avons utilisés pour faire l'analyse de la place de la mesure en génie logiciel. Ils nous ont permis d'y découvrir les incertitudes. Ces incertitudes ont, par la suite, constitué la base pour la formulation de nos questions sur la place de la mesure en génie logiciel.

Ainsi, à cette étape de notre recherche, nous étions prêts à identifier 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 13 du présent document.

3.2.3 Sondage sur les principes fondamentaux du génie logiciel

En plus de la réalisation des deux études Delphi sur les principes fondamentaux du génie logiciel, nous avions jugé nécessaire d'obtenir l'opinion d'un plus grand nombre de personnes en ce domaine afin de mieux valider nos données. Nous avons donc décidé de réaliser un sondage auprès de praticiens du génie logiciel Nous avons alors choisi d'interroger les membres du TCSE (Technical Committee on Software Engineering) de la IEEE Computer Society. La base de données des membres contenait près de 6000 entrées. Cependant, l'IEEE Computer Society ne détenait l'adresse électronique que d'environ 4000 de ces membres. Il fallait aussi s'attendre à ce que certaines adresses électroniques soient erronées. Toutefois, l'échantillon était malgré tout de taille intéressante. Nous avons donc décidé de procéder avec cet échantillon. Le sondage devait nous permettre également de récolter des données sur les opinions des praticiens du génie logiciel, en analysant des données sur 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 (voir appendice F). Ce sondage a été envoyé à près de 4000 praticiens en génie logiciel de l'IEEE.

Deux sondages effectués auparavant au sein de la IEEE Computer Society avaient obtenu un taux de réponse d'environ 17%. Nous nous attendions à un taux semblable pour notre sondage. 491 des 4000 adresses électroniques se sont révélées erronées. Sur un total d'approximativement 3500 envois, nous avons obtenu 574 réponses, ce qui constitue un taux de réponse de 16,4 pour cent. Ceci correspondait donc à nos attentes.

L'invitation à participer au sondage a été envoyée aux 4000 membres du TCSE par le truchement du courrier électronique. Cependant, le questionnaire a été placé sur le World Wide Web. Le message envoyé aux membres indiquait l'URL où l'on pouvait trouver la lettre d'introduction au sondage ainsi que le questionnaire lui-même. Un rappel émis quelques jours avant la date d'échéance du sondage nous a permis d'augmenter le nombre de réponses reçues.

Pour nous permettre de consulter les résultats en ligne, l'équipe technique de l'IEEE Computer Society, responsable du sondage, avait mis à notre disposition une page web. Dans cette page se trouvaient, groupées par colonnes, les réponses de chacun des participants. Il était aussi possible de cliquer sur le numéro d'un participant pour avoir accès à sa réponse au sondage. L'ensemble des résultats nous a ensuite été transmis sous forme de fichier Excel. C'est ce fichier que nous avons

remis au statisticien afin qu'il effectue les recherches dont la synthèse figure au chapitre présentant les résultats.

Afin de nous assurer de la qualité de notre sondage, nous avions 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 5 personnes a procédé à une seconde vérification. Avant d'être envoyé à ces quelques milliers de praticiens, le sondage a été envoyé, sous forme de pré-test à une cinquantaine de praticiens présentant les mêmes caractéristiques que les autres. Sur ces cinquante personnes, sept nous ont répondu et nous ont fait des commentaires pertinents, nous permettant d'améliorer le questionnaire du sondage. À la suite de chacune de ces vérifications, des changements ont été effectués, dans le but d'améliorer notre questionnaire. Ce questionnaire vous est présenté à l'annexe C de ce document.

3.2.4 Résumé de l'information recueillie à partir de la revue de littérature

Parallèlement à la conduite de nos différentes études sur les principes fondamentaux du génie logiciel, nous avons également effectué une revue de littérature. Nous avions pour objectif de consulter certains livres ainsi que des articles de colloques et de revues. Les sujets sur lesquels nous voulions nous concentrer étaient les suivants :

1. Le génie
   Nous voulions chercher à mieux comprendre le génie et à avoir une compréhension initiale du rôle qu'y joue la mesure.
2. Le génie logiciel
   Nous voulions 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 voulions 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 voulions tenter de comprendre la place de la mesure en génie logiciel.

Cette revue de littérature avait pour objectifs de nous donner une meilleure compréhension de notre sujet et de nous permettre de présenter des arguments solides. Nous nous sommes attardés à comprendre le rôle de la mesure dans le génie logiciel. Pour ce faire, nous avons utilisé 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)

3.3 MÉTHODE DE TRAITEMENT DES DONNÉES

Nous voulons appliquer des tests statistiques très simples, tels la moyenne, 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 avions cherché 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 avons utilisé des méthodes statistiques pour faire ressortir les corrélations. Le sondage nous a permis 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 a été envoyé à près de 4000 praticiens en génie logiciel de l'IEEE. 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 a été 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 effectués, dans le but d'améliorer notre questionnaire. Ce questionnaire vous est présenté à l'annexe C de ce document.

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

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 n'avons retenu que les commentaires exprimant une opinion sur les principes car ces derniers nous permettent 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 avons éliminé ceux ne se rapportant pas à la mesure. Pour ceux qui se rapportent à la mesure, nous les avons classifiés selon le type de mesure, tel que défini dans notre cadre conceptuel. Les catégories sont 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 n'ont pus être classés selon ces catégories car ils concernaient la mesure d'un point de vue général ou encore le quoi et le comment mesurer. Ils se rapportaient probablement également à la question de l'utilité même de la mesure en génie logiciel. Nous nous sommes intéressés aussi à eux puisqu'ils nous permettent d'avoir une section du questionnaire portant sur la nécessité de mesurer en génie logiciel. Cette section est importante car, avant de savoir quoi mesurer, il faut savoir s'il est nécessaire de mesurer.

Nous avons ensuite effectué une synthèse de ces données. Le but de cette synthèse était d'identifier les incertitudes de la mesure en génie logiciel que les participants avaient mentionnés dans les commentaires. Lors de cette synthèse, tous les commentaires retenus à l'étape précédente ont été analysés pour y chercher les leçons, critiques, ou autres, qui y étaient contenues. Nous avons identifié les commentaires pouvant être regroupés. Nous avons également identifié les nuances entre ces commentaires. Puis, nous avons analysé 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 a consisté à décider des sujets sur lesquels portaient les questions.

Une fois la synthèse de données terminée, nous sommes passée 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 ont également effectué une vérification de l'analyse des données qualitatives.

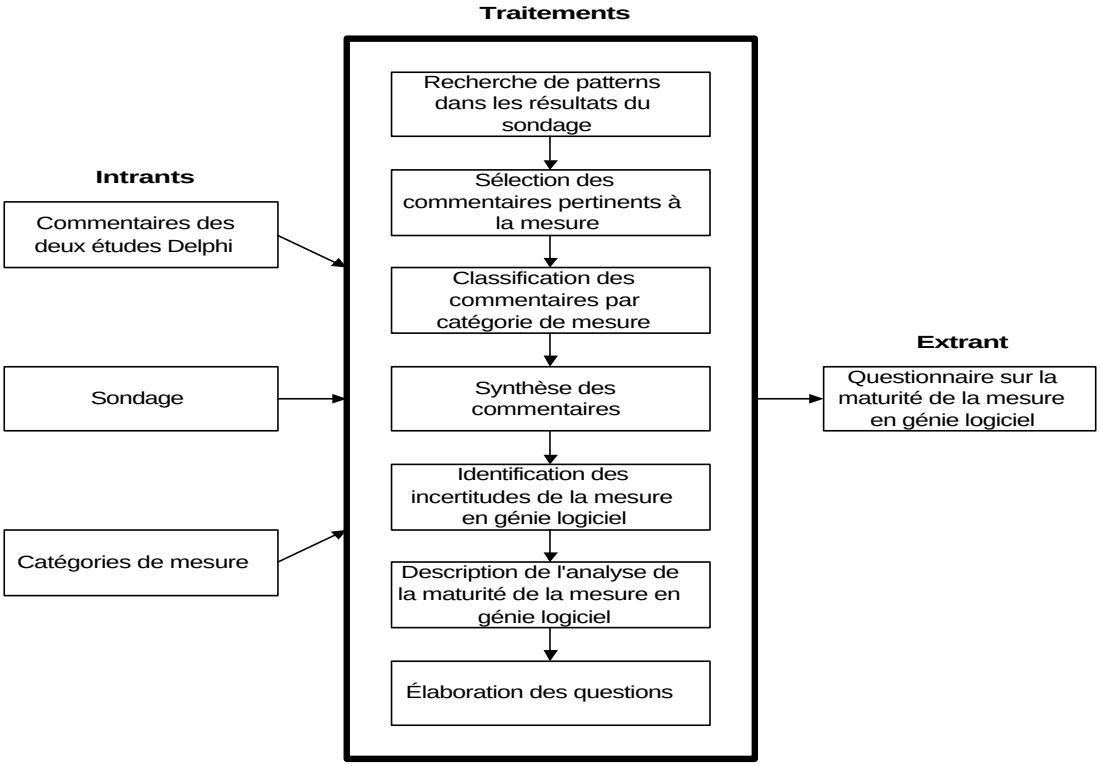

Figure 3.3 Étapes de production du questionnaire sur la mesure en génie logiciel

4. PRÉSENTATION DES DONNÉES QUANTITATIVES RECUEILLIES

Dans cette section, nous vous présentons d'abord la liste des principes candidats du génie logiciel que nous avons retenus. Puis, pour chacune des trois études ayant permis d'aboutir à ces principes et de les évaluer, nous vous présentons une synthèse des résultats obtenus. Nous effectuons également une présentation synthétisée des résultats de ces trois études.

4.1 PRINCIPES FONDAMENTAUX DU GÉNIE LOGICIEL

Dans cette section, nous vous présentons la liste de quinze principes fondamentaux du génie logiciel. Le principe A est relié à la mesure. La plupart des commentaires s'y rapportant soint 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 certains commentaires s'y rapportant parlaient également de mesure. On peut citer, en particulier, les principes K et O.

A. Appliquer et utiliser des mesures quantitatives dans la prise de décision
B. Construire en réutilisant et pour être réutilisé
C. Contrôler la complexité de par multiples perspectives et multiples niveaux
D. Définir les artefacts logiciels de façon rigoureuse
E. Mettre en place un processus logiciel flexible
F. Implanter un processus rigoureux et l'améliorer continuellement
G. Consacrer les ressources nécessaires à la compréhension du problème
H. Gérer d'une manière aussi formelle que possible la qualité durant l'ensemble du cycle de vie
I. Minimiser les interactions entre les composants logiciels
J. Produire le logiciel par étapes
K. Établir des objectifs de qualité pour chaque produit à livrer
L. Comme le logiciel est, de par sa nature même, sujet au changement, il faut planifier et gérer ce changement
M. Les compromis étant inhérents au génie logiciel, il faut les rendre explicites et les documenter
N. Pour améliorer la conception, étudier les solutions antérieures à des problèmes similaires
O. L'incertitude est inévitable en génie logiciel. Il faut identifier et gérer cette incertitude

4.2 RÉSULTATS DE LA PREMIÈRE ÉTUDE DELPHI

Tableau 4.1

Synthèse des résultats de la première étude Delphi

PRINCIPE	MOYENNE	CONSENSUS
A	7,60	7
B	8,00	7
C	N.A.	N.A.
D	6,40	6
E	7,60	7
F	6,90	4
G	8,70	7
H	7,80	7
I	7,30	8
J	7,70	7
K	7,70	8
L	9,10	9
M	8,40	8
N	N.A.	N.A.
O	8,00	8

Le tableau 4.1 présente la liste des principes, les moyennes (allant de 1 à 10) ainsi que le nombre de personnes se ralliant au score obtenu par chaque principe. Nous constatons que les scores moyens alloués à chaque principe varient entre 9,1 et 6,4 sur un maximum de 10. En règle générale, les principes ayant obtenu les scores les plus élevés sont ceux pour lesquels le consensus est le plus élevé.

Les principes L[6] et G[7] sont les plus populaires selon la première étude Delphi. Ils ont obtenu respectivement des moyennes de 9,1 et de 8,7 sur 10 et un consensus de 12 et de 10 sur 12.

Dans la première étude Delphi, le principe A, portant sur les mesures quantitatives, a obtenu une moyenne de 7,6 sur 10, ce qui le classe à la neuvième position dans la liste des scores moyens. Par contre, le consensus pour le principe A est de 9 sur 12, ce qui signifie que 9 personnes sur 12 pensent que le score de 7,6 sur 10 convient à ce principe.

---

6 Principe L : *Comme le logiciel est, de par sa nature même, sujet au changement, il faut planifier et gérer ce changement*

7 Principe G : *Consacrer les ressources nécessaires à la compréhension du problème*

4.3 RÉSULTATS DE LA DEUXIÈME ÉTUDE DELPHI

Tableau 4.2

Synthèse des résultats de la deuxième étude Delphi

PRINCIPE	MÉDIANE	CONSENSUS
A	7	13
B	9	17
C	8	23
D	8	22
E	8	21
F	8	19
G	10	29
H	9	20
I	9	25
J	8	23
K	8	20
L	10	26
M	9	25
N	9	24
O	10	25

Le tableau 4.2 présente la liste des principes, les médianes (allant de 1 à 10) ainsi que le nombre de personnes se ralliant au score obtenu par chaque principe. Nous constatons que les scores médians alloués à chaque principe varient entre 10 et 7 sur un maximum de 10. En règle générale, les principes ayant obtenu les scores les plus élevés sont ceux pour lesquels le consensus est le plus élevé.

Pour la deuxième étude Delphi, ce sont les principes G[8], L[9] et O[10] qui ont obtenu les médianes les plus élevées (une médiane de 10 sur 10 pour les trois principes) avec, respectivement, des consensus

[8] Principe G : *Consacrer les ressources nécessaires à la compréhension du problème*

de 29, 26 et 25 sur 29. On se rend compte que, en ce qui concerne les principes G et L, les résultats des deux études Delphi concordent assez bien.

Dans la deuxième étude Delphi, le principe A, portant sur les mesures quantitatives, a obtenu une médiane de 7 sur 10, ce qui le classe comme étant le principe ayant la médiane la plus faible. Par contre, le consensus pour le principe A est de 13 sur 29, ce qui montre que les participants n'étaient pas réellement d'accord avec le score obtenu par ce principe.

---

[9] Principe L : *Comme le logiciel est, de par sa nature même, sujet au changement, il faut planifier et gérer ce changement*

[10] Principe O : *L'incertitude est inévitable en génie logiciel. Il faut identifier et gérer cette incertitude*

4.4 RÉSULTATS DU SONDAGE

Tableau 4.3

Synthèse des résultats du sondage

PRINCIPE	MÉDIANE	ÉCART-TYPE
A	8	2,42
B	8	2,34
C	8	2,44
D	8	2,52
E	8	2,28
F	9	2,36
G	10	2,05
H	8	2,49
I	7	2,69
J	7	2,66
K	8	2,32
L	9	2,08
M	9	2,31
N	8	2,17
O	8	2,48

Le tableau 4.3 présente la liste des principes, les médianes (allant de 1 à 10) ainsi que le nombre de personnes se ralliant au score obtenu par chaque principe. Nous constatons que les scores médians alloués à chaque principe varient entre 10 et 7 sur un maximum de 10. En règle générale, les principes ayant obtenu les scores les plus élevés sont ceux pour lesquels l'écart-type est le moins grand. L'écart-type pour les différents principes varie de 2,05 à 2,69.

Pour le sondage, c'est le principe G qui a obtenu la plus forte médiane, c'est-à-dire 10 sur un maximum de 10. Trois autres principes ont obtenu une médiane de 9. Ce sont les principes F[11], L[12] et M[13].

Dans le sondage, le principe A, portant sur les mesures quantitatives, a obtenu une médiane de 8 sur 10. La plus forte médiane était de 10 mais il est difficile de classer le principe A par rapport aux autres parce que huit autres principes ont également une médiane de 8. L'écart-type est de 2,42 pour ce principe.

---

11 Principe F : *Implanter un processus rigoureux et l'améliorer continuellement*

12 Principe L : *Comme le logiciel est, de par sa nature même, sujet au changement, il faut planifier et gérer ce changement*

13 Principe M : *Les compromis étant inhérents en génie logiciel, il faut les rendre explicites et les documenter*

4.5 PRÉSENTATION DES RÉSULTATS DES TROIS ÉTUDES

Tableau 4.4

Synthèse des scores pour les trois études sur les principes fondamentaux

PRINCIPE	MOYENNE DELPHI I	MÉDIANE DELPHI II	MÉDIANE SONDAGE
A	7,60	7	8
B	8,00	9	8
C	N.A.	8	8
D	6,40	8	8
E	7,60	8	8
F	6,90	8	9
G	8,70	10	10
H	7,80	9	8
I	7,30	9	7
J	7,70	8	7
K	7,70	8	8
L	9,10	10	9
M	8,40	9	9
N	N.A.	9	8
O	8,00	10	8

Au cours des trois études, les principes qui ressortent comme étant les plus populaires sont le G et le L. Le principe G parle d'investir dans la compréhension du problème. Bien des participants ont même dit que c'était quelque chose de trop évident pour être un principe. Cependant, les expériences passées en informatique et en génie logiciel montrent que l'étape de compréhension du problème ne reçoit pas toujours l'attention qui lui revient. C'est probablement pourquoi, au bout du compte, ce principe a été le plus populaire de tous. Le principe L dit ceci : puisque les changements sont inhérents au génie logiciel, il est nécessaire de planifier et de gérer en fonction de ces changements.

Certains disent que ce n'est pas vrai que les changements sont inhérents au génie logiciel. D'autres disent qu'il y a

Tableau 4.5

Synthèse des consensus pour les trois études sur les principes fondamentaux

PRINCIPE	CONSENSUS DELPHI I	CONSENSUS DELPHI II	ÉCART-TYPE SONDAGE
A	7	13	2,42
B	7	17	2,34
C	N.A.	23	2,44
D	6	22	2,52
E	7	21	2,28
F	4	19	2,36
G	7	29	2,05
H	7	20	2,49
I	8	25	2,69
J	7	23	2,66
K	8	20	2,32
L	9	26	2,08
M	8	25	2,31
N	N.A.	24	2,17
O	8	25	2,48

toujours des changements partout. Ce qui nous paraît essentiel dans ce principe, c'est la nécessité de planifier et de gérer en fonction de ces changements.

Le principe A, celui concernant l'utilisation de mesures quantitatives, n'a pas eu de résultats aussi bons. Les résultats, particulièrement ceux de la deuxième étude Delphi, semblent indiquer que les participants ne s'entendent pas sur l'importance de ce principe. Au début de notre document, nous mentionnions que si le génie logiciel doit faire partie du génie, alors la mesure doit y jouer un rôle

important. Ainsi, ces résultats suggèrent l'une des possibilités suivantes : soit le génie logiciel, de par son immaturité, utilise mal et de façon insuffisante la mesure, soit ce n'est pas du génie.

Les deux autres principes dont certains commentaires sont reliés à la mesure sont les principes K et O. Le principe K dit d'établir des objectifs de qualité pour chaque produit à livrer. Les participants ont souvent mentionné la nécessité de pouvoir mesurer l'atteinte de ces objectifs. Au cours des trois études, ce principe a obtenu des scores plutôt faibles. Au point de vue du consensus, les participants de la deuxième étude Delphi se ralliaient davantage au score que ceux de la première étude Delphi. Le principe O parle d'incertitude et dit que c'est quelque chose d'inévitable en génie logiciel. Certains participants parlaient du rôle de la mesure pour aider à diminuer cette incertitude. Ce principe a été jugé fondamental, avec un très fort consensus, lors de la deuxième étude Delphi, ce qui n'est pas le cas pour les deux autres études. Nous tenons à faire remarquer que ces résultats ne reflètent pas la place de la mesure au sein de ces principes, ce qui se retrouve plutôt au niveau des commentaires des participants. De ce fait, nous ne faisons que mentionner les résultats obtenus pour les principes ayant un certain rapport avec la mesure.

5. ANALYSE DE LA PLACE DE LA MESURE AU SEIN DU GÉNIE LOGICIEL

Dans la section précédente, nous avons analysé les résultats quantitatifs des trois études sur les principes fondamentaux. Dans la présente section, nous abordons l'analyse des commentaires faits par les participants lors de ces trois études.

Nous réalisons cette analyse de la mesure en génie logiciel en procédant de la manière suivante :

1) les commentaires liés à la mesure sont regroupés selon le sujet et synthétisés en quelques lignes,
2) nous faisons un lien avec la revue de la littérature
3) nous identifions des incertitudes de la mesure en génie logiciel

Les incertitudes sont tirées principalement des commentaires des études Delphi et du sondage. Rappelons que seuls les commentaires liés à la mesure sont traités ici et qu'ils sont classés en fonction des sept raisons de mesurer identifiées dans la revue de littérature en plus d'une catégorie portant sur la mesure en général. Vous pouvez retrouver les commentaires liés à la mesure à l'annexe E du présent document. Ces commentaires nous proviennent des participants aux trois études mentionnées ci-dessus. À la fin de chaque paragraphe sur les différents sujets, nous établissons un lien avec ce qui a été retenu de la revue de la littérature.

Dans les paragraphes qui suivent, nous avons choisi de mettre un titre pour chaque groupe de commentaires, en fonction, du sujet sur lequel ils portent. Bien sûr, tous ces sujets se rattachent au génie logiciel. Cependant, dans le but d'alléger les titres, nous avons omis de placer cette expression dans chacun des titres.

5.1 MESURE EN GÉNÉRAL

L'importance de la mesure dans la reconnaissance du génie logiciel en tant que discipline du génie

Commentaire(s) : 50, 52, 71, 85

La mesure joue un rôle extrêmement important à l'intérieur de toutes les disciplines du génie. C'est pourquoi, selon certains participants, si la mesure n'est pas utilisée au sein du génie logiciel, alors cette discipline pourrait ne pas être acceptée en tant que discipline du génie par les autres disciplines en faisant partie. Toutefois, un participant a fait remarquer que, malgré l'importance qu'a la mesure, la pratique du génie logiciel restera toujours une combinaison d'art et de science. Cela nous amène à parler de deux incertitudes. La première est de savoir si la mesure joue un rôle primordial en génie logiciel. La deuxième est que si la mesure se révélait n'être pas essentielle en génie logiciel, cela éliminerait-il les chances de cette discipline d'être reconnue par différents organismes comme branche du génie. Fenton et Pfleeger (1997)[14] nous parlent de l'importance de la mesure en génie et de sa négligence en génie logiciel. Ils disent qu' « il est difficile d'imaginer le génie électrique, mécanique ou civil sans un rôle central pour la mesure » tandis que « la mesure a été considérée comme un luxe en génie logiciel (…) Quand on prend des mesures, on le fait habituellement de manière peu fréquente, inconsistance et incomplète ».

- La mesure joue-t-elle un rôle primordial en génie logiciel?
- Si la mesure ne joue pas un rôle primordial en génie logiciel, cela élimine-t-il les chances de cette discipline d'être reconnue par différents organismes comme branche du génie?

La nécessité d'utiliser des mesures

Commentaire(s) : 18, 79

Un participant a eu à dire que l'application et l'utilisation des mesures quantitatives dans la prise de décision était souhaitable mais pas toujours pratique. Nous pouvons toujours nous questionner sur le fait que prendre des mesures est pratique ou pas en génie logiciel. Par exemple, un autre participant a suggéré de mettre l'attention sur les habiletés et la formation continue des personnes qui, d'ailleurs, représentent les atouts capitaux les plus critiques, plutôt que sur la prise de mesures quantitatives. Cependant, nous pouvons également aller plus loin et nous demander si, parce que ce n'est pas toujours pratique, faut-il ne pas prendre de mesures? Fenton et Pfleeger (1997), dans leur livre Software Metrics, nous donnent l'exemple de la température ambiante, du temps, etc. qui étaient des entités impossibles à mesurer il y a quelques siècles de cela. Pourtant, nous avons appris à mieux connaître ces entités au point d'en arriver à trouver des moyens de mesurer leurs attributs. Fenton et

[14] Toutes les références au livre de Fenton et Pfleeger (1997) concernent uniquement le contenu

Pfleeger donnent ces exemples afin de nous démontrer que les entités du génie logiciel nous sont inconnues, mais qu'il faut apprendre à mieux les connaître afin d'être un jour à même de les mesurer. Ils citent d'ailleurs Galileo Galilei qui dit de rendre mesurable ce qui ne l'est pas. Alors, même s'il n'est pas pratique d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision en génie logiciel, il serait bon de savoir s'il faut malgré tout tenter de mesurer les attributs de ces entités. L'incertitude que nous identifions ici est de savoir s'il est nécessaire de mesurer les attributs de certaines entités en génie logiciel, même si nous ne connaissons pas très bien ces entités. Est-ce que ces mesures sont utiles à quelque chose? Autrement dit, quels avantages nous apporte l'utilisation de mesures en génie logiciel? Fenton et Pfleeger se questionnent également sur une série de thèmes en rapport avec la nécessité de mesurer en génie logiciel. Certaines des questions qu'ils se posent sont les suivantes : Quelle quantité d'information faut-il avoir sur un attribut avant de considérer qu'il est raisonnable de le mesurer? Comment savoir si on a réellement mesuré l'attribut que l'on voulait mesurer?

- Est-ce parce qu'il n'est pas toujours pratique de prendre des mesures en génie logiciel qu'il ne faut pas en prendre?
- Est-ce nécessaire de mesurer certaines entités en génie logiciel si nous ne connaissons pas très bien ces entités?
- Quels avantages nous apporte l'utilisation de mesures en génie logiciel?
- Quelle quantité d'information faut-il avoir sur un attribut avant de considérer qu'il est raisonnable de le mesurer?
- Comment savoir si on a réellement mesuré l'attribut que l'on voulait mesurer?

Quoi mesurer?

Commentaire(s) : 1, 74

Une question de base concernant la mesure en génie logiciel est de savoir quoi mesurer. L'un des participants a suggéré de mesurer ce qui importe réellement et non pas ce qui est facile à mesurer. Cela constitue déjà un premier critère permettant de savoir quoi mesurer. Fenton et Pfleeger (1997) sont aussi de cet avis. Eux aussi disent de mesurer ce qui est réellement nécessaire plutôt que ce qu'il est commode ou facile de mesurer. Grady (1994) dit qu'il est difficile de réduire ce qu'on doit mesurer à un petit ensemble de mesures pour les gestionnaires de haut niveau. Par contre, il suggère une liste de données à collecter. Un autre participant suggérait de mesurer les connaissances des

---

des trois premiers chapitres de ce livre.

individus et des équipes dans la production de logiciel. Il suggérait aussi de mesurer l'impact que peuvent avoir les artefacts logiciels sur l'économie et l'industrie. Ces suggestions montrent que ce qu'on peut mesurer est très diversifié. Cette question de savoir quoi mesurer demeure donc incertaine. Nous nous demandons alors quoi mesurer en génie logiciel?

- Quoi mesurer en génie logiciel?

Sur quoi est basée la prise de décision?

Commentaire(s) : 58, 72, 75, 84

Dans l'un des commentaires, un participant s'est demandé sur quoi baser la prise de décision si ce n'est pas sur des mesures quantitatives. Quelques uns proposent des choses telles que l'intuition, les preuves, le « gut feeling » et l'expérience d'utilisation. Le participant qui s'est posé cette question a proposé comme réponse l'intuition, les croyances et a ajouté que puisque l'utilisation de ces derniers dans la prise de décision aura lieu malgré tout, autant essayer de baser le plus possible les décisions sur des mesures quantitatives. Ceci génère l'incertitude suivante : puisque l'intuition et les croyances seront probablement utilisées dans la prise de décision en génie logiciel, pourquoi ne pas essayer plutôt de baser cette prise de décision sur des données quantitatives? Cependant, un autre participant a suggéré qu'il faudrait plutôt utiliser les mesures quantitatives comme simple assistant dans la prise de décision car elles peuvent parfois donner de mauvaises indications. Il a ajouté que la prise de décision était un processus humain et que les mesures étaient simplement des outils à utiliser dans la prise de décision. En ce qui concerne l'utilisation de l'intuition dans la prise de décision, Fenton et Pfleeger (1997) pensent qu'elle devrait constituer le point de départ pour toutes les mesures, parce que le but de la théorie représentative de la mesure est de formaliser notre intuition à propos du mode de fonctionnement du monde.

- Puisque l'intuition et les croyances seront probablement utilisées dans la prise de décision en génie logiciel, ne faut-il pas plutôt essayer de baser cette prise de décision sur des données quantitatives?

En savons-nous assez pour être quantitatif?

Commentaire(s) : 59, 82

Un autre argument retrouvé dans les commentaires et concernant la mesure soutient que nous n'en savons pas assez pour être quantitatif. L'auteur de ce commentaire ajoute que nous avons de la difficulté à convaincre les gens de la nécessité de bien comprendre le problème. Autrement dit, si on ne peut pas s'attarder à comprendre le problème, on s'attardera encore moins à baser la prise de décision sur des mesures quantitatives ou bien sinon ce sera une tâche impossible à accomplir. Un autre participant a ajouté que le principe A constituait un but élevé que nous ne comprenions pas bien. Ça nous amène alors à ces deux incertitudes : avons-nous une compréhension suffisante du domaine du génie logiciel pour nous permettre d'être quantitatif? Ne faut-il pas arriver à maîtriser des aspects de base du génie logiciel, telle la compréhension du problème, avant de s'aventurer à utiliser les mesures quantitatives dans la prise de décision?

- Avons-nous une compréhension suffisante du domaine du génie logiciel pour nous permettre d'être quantitatif?
- Ne faut-il pas arriver à maîtriser des aspects de base du génie logiciel, telle la compréhension du problème, avant de s'aventurer à utiliser les mesures quantitatives dans la prise de décision?

Maturité et fiabilité des mesures

Commentaire(s) : 29

Le point de vue d'un des participants concernant les mesures quantitatives ressemble un peu à celui du paragraphe précédent. Le participant a dit qu'il ne pensait pas que celles-ci soient ni assez matures ni fiables dans la prise de décision. Il pense que les gestionnaires de génie logiciel devraient avoir des critères aussi bons que ceux utilisés dans la gestion des autres activités d'ingénierie. Cependant, pour parvenir à avoir ces critères, ne faut-il pas justement tenter de mesurer certains attributs des entités afin de mieux les connaître? Le participant a également parlé de maturité et de fiabilité, ce qui nous amène à identifier l'incertitude suivante : les mesures en génie logiciel sont-elles réellement immatures et non fiables?

- Pour que les gestionnaires de génie logiciel arrivent à avoir des critères aussi bons que ceux utilisés dans la gestion des autres activités d'ingénierie, ne faut-il pas justement tenter de mesurer certains attributs des entités afin de mieux les connaître?
- Les mesures en génie logiciel sont-elles réellement immatures et non fiables

Le qualitatif dans la prise de décision

Commentaire(s) : 1, 14, 20, 23, 25, 27, 28, 30, 32, 34, 37, 38, 55, 57, 71, 72

Certains pensent qu'il ne faut pas négliger le côté qualitatif dans la prise de décision en génie logiciel. Des commentaires conseillent d'utiliser le quantitatif lorsque c'est possible et pratique mais sinon d'utiliser le qualitatif. Certains pensent même que le qualitatif peut avoir plus d'importance que le quantitatif. D'autres pensent également qu'il ne faut surtout pas que le quantitatif soit totalement exclu. Ce qui ressort surtout de ces commentaires, c'est que le qualitatif ne devrait absolument pas être exclu. L'incertitude que nous identifions ici est que nous ne savons pas si le quantitatif est plus important que le qualitatif et aussi quand, pourquoi et comment il faut utiliser le quantitatif. Fenton et Pfleeger (1997) parlent également des mesures qualitatives, basées habituellement sur le jugement de certains individus. Ils donnent comme exemple le vin dont la qualité est mesurée par des experts et se demandent si ce genre de mesures est juste. Ils rajoutent que les mesures subjectives peuvent être utiles tant qu'on admet leur imprécision.

- Le qualitatif a-t-il plus d'importance que le quantitatif?
- Quand, pourquoi et comment faut-il utiliser le quantitatif?

Le degré d'importance du quantitatif dans la prise de décision

Commentaire(s) : 56, 81

Un participant a affirmé qu'il était essentiel d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision. En parlant du principe A, un autre participant rajoute que si ce dernier implique que la prise de décision ne repose pas uniquement sur des mesures quantitatives, alors il lui donnera un score de 10. L'incertitude que nous avons ici est une question de base en génie logiciel et certains diront même que si ce n'est pas fait, alors que le génie logiciel n'est simplement pas du génie. Cette incertitude consiste à savoir s'il est essentiel d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision.

- Est-il essentiel d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision?

Mesures quantitatives de la qualité

Commentaire(s) : 21, 43, 73

L'un des principes fondamentaux énonce ceci : gérez la qualité tout au long du cycle de vie de manière aussi formelle que possible. L'un des participants a ajouté à cet énoncé qu'il fallait également le faire de manière aussi quantitative que possible. Cependant, dans de nombreux commentaires, on mentionnait le fait que la qualité ne soit pas bien définie et donc qu'elle peut avoir une signification différente pour chaque personne. Ainsi, nous nous posons la question suivante : Peut-on mesurer la qualité en génie logiciel si ce terme n'y est pas bien défini? Cela revient à ce qu'affirmaient Fenton et Pfleeger (1997) : qu'il faut connaître une entité et ses attributs avant de pouvoir la mesurer. Ces mêmes auteurs trouvent d'ailleurs une éventuelle explication à la difficulté de mesurer la qualité en évoquant le côté multi-dimensionnel de la qualité qui ne reflète pas un aspect unique d'un produit.

Peut-on mesurer la qualité en génie logiciel si ce terme n'y est pas bien défini?

Mesurer les objectifs de qualité

Commentaire(s) : 3, 26, 44

Le principe K conseille d'établir des objectifs de qualité pour chaque produit livrable. L'un des participants a ajouté qu'il est cependant bien plus crucial de les établir de façon à pouvoir les mesurer et les tester. Cela nous ramène à ce que nous mentionnions plus haut, à savoir que certains participants pensent que la qualité en génie logiciel n'est pas clairement définie. Cependant, le problème se présente sous un autre angle ici car le principe fondamental dit d'établir ces objectifs de qualité. Il faudrait alors se demander ceci : s'il faut établir des objectifs de qualité pour chaque produit livrable, faut-il le faire de manière à pouvoir les mesurer? Sont-ils mesurables? Est-il réaliste de fixer des objectifs de qualité si on ne peut pas les mesurer? De plus, l'un des participants a mentionné qu'il fallait une organisation très mature pour établir des objectifs de qualité. Cela veut-il dire que seul les organisations ayant atteint un certain degré de maturité peuvent établir de tels objectifs? Qu'en est-il des organisations moins matures? N'y a-t-il pas moyen malgré tout pour de telles organisations d'arriver à se fixer des objectifs de qualité?

- S’il faut établir des objectifs de qualité pour chaque produit livrable, faut-il le faire de manière à pouvoir les mesurer?
- Peut-on mesurer de tels objectifs?
- Est-ce que seul les organisations plus matures peuvent établir des objectifs de qualité?
- N’y a-t-il pas moyen malgré tout pour des organisations moins matures d’arriver à se fixer des objectifs de qualité?

Quand faut-il utiliser des mesures quantitatives dans la prise de décision?

Commentaire(s) : 60

Le principe A dit d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision. Cependant, l'un des participants a déclaré que ceci n'est ni applicable universellement, ni désirable universellement. Cela voudrait dire qu'il y aurait des cas où il serait préférable de ne pas utiliser ces mesures quantitatives. L'incertitude ici est que nous ne savons pas s'il faut toujours utiliser les mesures quantitatives dans la prise de décision. De plus, si la réponse est non, comment savoir quand appliquer ces mesures?

- Faut-il toujours utiliser des mesures quantitatives dans la prise de décision?
- De plus, si la réponse est non, comment savoir quand appliquer ces mesures?

Utilité des mesures «quantitatives» dans la prise de décision

Commentaire(s) : 36

Un participant a affirmé être d'accord avec le principe : Appliquez et utilisez des mesures dans la prise de décision. C'est-à-dire qu'il laisserait tomber le mot «quantitatif» du principe. Il a dit, par exemple, qu'un choix d'outils à utiliser pourrait être basé sur une étude de marché plutôt que sur une analyse technologique (scientifique). De plus, il pense que la gestion de logiciels et les décisions d'implantation devraient incorporer les objectifs d'affaire, d'entreprise, de produit et de technologie de l'organisation. Il baserait plutôt la prise de décision sur une analyse complète des faits, ceux-ci pouvant être aussi bien des opinions générées par la technique Delphi qu'une analyse statistique de données. Ce qui différencie donc le principe proposé du principe A, c'est l'utilisation de mesures quantitatives dans la prise de décision. Pourtant, même le participant parle d'analyse de données statistiques, ce qui implique déjà du quantitatif. L'incertitude que nous relevons ici est alors la suivante : l'utilisation de données uniquement non quantitatives suffit-elle pour la prise de décision ou faut-il absolument y ajouter des mesures quantitatives?

- L'utilisation de données uniquement non quantitatives suffit-elle pour la prise de décision ou faut-il y ajouter des mesures quantitatives?

Utilisation de la mesure pour identifier et gérer l'incertitude

Commentaire(s) : 69, 70, 77

Les commentaires que nous traitons ici se rattachent au principe O. Ce dernier énonce ceci : Puisque l'incertitude est inévitable en génie logiciel, identifiez-la et gérez-la. Les auteurs de ces commentaires recommandent de combiner ce principe avec le principe A qui, nous le rappelons, suggère d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision. Ces auteurs suggèrent de soupeser les incertitudes en se basant sur la collecte de données. L'un des participants souhaite même que ces faits soient quantitatifs. Nous pensons également que cette collecte de faits implique la nécessité d'utiliser des mesures, au moins jusqu'à un certain point. La première question que nous nous posons ici est de savoir si l'utilisation de mesures quantitatives peut réellement permettre d'identifier les incertitudes? Puis, nous nous demandons si, une fois les incertitudes identifiées, les mesures peuvent nous aider à mieux les gérer? De plus, nous cherchons à savoir si la mesure pourrait indirectement faciliter la prise de décision, si elle permettrait réellement de diminuer le niveau d'incertitude dans les projets?

- L'utilisation de mesures quantitatives peut-elle réellement permettre d'identifier les incertitudes?
- Une fois les incertitudes identifiées, les mesures peuvent-elles nous aider à mieux les gérer?
- La mesure pourrait-elle indirectement faciliter la prise de décision si elle permettrait réellement de diminuer le niveau d'incertitude dans les projets?

La validité des mesures

Commentaire(s) : 13 et 16, 76

Un commentaire conseillait de faire attention aux nombres qui sont faux ou tout fabriqués. Un autre énonçait qu'aucun estimé n'est correct, sauf par chance. La mesure peut probablement permettre d'améliorer la justesse des estimés. Selon Grady (1994), ce serait effectivement le cas. Il parle d'utiliser des données de projets passés pour vérifier les estimés. Ces remarques portent également à se questionner sur la validité des mesures utilisées en génie logiciel. En effet, il ne suffit pas de mesurer pour obtenir un nombre. Il faut aussi que les résultats de la mesure représentent quelque chose qui pourra aider à l'avenir à améliorer le développement de logiciels. Nous pensons que c'est

un problème réel en génie logiciel car beaucoup d'articles parlent de validité des outils ou méthodes de mesures dans ce domaine. Des mesures telles que le nombre de lignes de code ou les points de fonction sont souvent utilisées en génie logiciel mais reçoivent cependant beaucoup de critiques quant à leur valeur réelle. Fenton et Pfleeger (1997) pensent qu'il y a un réel manque de validation des mesures en génie logiciel. Ils nous conseillent d'ailleurs de ne pas se presser de générer des nombres sans considérer leur signification réelle. Ils mentionnent aussi deux erreurs souvent faites concernant la validation des données : celle de compter sur la crédibilité de celui qui propose la mesure et celle de vérifier la validité d'une mesure en la comparant à une mesure déjà existante sans savoir si cette dernière est elle même valide.

Deux autres facteurs pouvant également affecter la validité des résultats des mesures sont décrits dans l'article de Weller « Using Metrics to Manage Software Projects ». Ce sont la validité et l'absence des données. En effet, si les données de base sont erronées, il est difficile d'obtenir des résultats fiables. D'un autre côté, si les données ne sont pas encore disponibles, on ne peut pas fournir de résultat. Cela représente une autre incertitude, celle de savoir si, dans de nombreux cas, nous n'avons pas affaire à des données erronées ou à un manque de données? Fenton et Pfleeger (1997) le disent aussi en mentionnant que la « qualité d'un programme de mesure dépend clairement d'une collecte attentive des données ». Cependant, ils pensent aussi que c'est quelques chose de plus facile à dire qu'à faire. Dans l'article « Using Metrics in Management Decision Making », Stark et Durst disent que les métriques ne doivent pas nécessairement être exacts pour être utiles. Cela nous semble un peu invraisemblable. C'est pourquoi nous posons aussi les questions suivantes : Est-ce que des mesures inexactes peuvent être utiles en génie logiciel et, si oui, que doit être la marge d'erreur à ne pas dépasser?

Dans la revue de littérature, nous avons découvert une autre incertitude concernant la validité des données, celle de savoir si les mesures accomplissent réellement les tâches qui leur sont assignées. En d'autres mots, nous nous demandons à quel point la validité interne des mesures utilisées en génie logiciel est respectée, celle-ci étant définie comme le degré de confiance qu'on a en la véracité des résultats pour une situation d'étude spécifique (Davis, 1996)? Nous cherchons à connaître le degré de validité interne car, selon Davis, ce type de validité n'est jamais complètement mesurable. La raison en est que les sources d'invalidité sont trop nombreuses pour que le chercheur les mesure et même les identifie toutes, dans l'environnement de recherches d'affaires. Selon Basili, Selby et Hutchens (1994), la validité interne des mesures ne serait pas toujours respectée en génie logiciel.

D’autres questions concernant la validité interne des données sont posées par Fenton et Pfleeger (1997). Ce sont les suivantes : En utilisant les mesures, quelle affirmation sensée peut on faire à propos des attributs et des entités qui les possèdent (par exemple, est-il sensé de parler de doubler la qualité de la conception)? Quelles opérations sensées peut-on effectuer sur les mesures (par exemple, peut-on calculer la productivité moyenne d’un groupe de développeurs)? Par « opération sensée », Fenton et Pfleeger entendent qu’il est nécessaire de savoir à quel type d’échelle de mesure (nominal, ordinal, interval, ratio, absolu ou autres) appartient une entité et qu’il est impératif d’utiliser uniquement les opérations statistiques permises avec ce genre d’échelle.

Puisque nous parlons de validité interne, nous pouvons également aborder la question de validité externe des mesures. Celle-ci est définie comme le degré selon lequel les résultats d’une étude peuvent être généralisés à travers les populations, les contextes et d’autres conditions similaires (Davis, 1996). Cette question est étroitement liée à la précédente car Davis nous dit qu’il ne peut y avoir de validité externe sans validité interne?

- La mesure peut-elle nous aider à améliorer la justesse des estimés?
- Les outils et les méthodes de mesure utilisés en génie logiciel sont-ils valides?
- À quel point la validité interne des mesures en génie logiciel est-elle respectée?
- À quel point peut-on parler de validité externe des données en génie logiciel?
- Est-ce que des mesures inexactes peuvent être utiles en génie logiciel et, si oui, que doit être la marge d’erreur à ne pas dépasser?
- En utilisant les mesures, quelle affirmation sensée peut on faire à propos des attributs et des entités qui les possèdent?
- Quelles opérations sensées peut-on effectuer sur les mesures?

Les coûts versus les bénéfices (ou l'apport) de la mesure

Commentaire(s) : 1, 34, 42, 51, 77, 83

Certains participants ont émis des commentaires selon lesquels il faut évaluer l'apport de la mesure versus les coûts rattachés à la collecte et à l'analyse de l'information. Si la valeur qu'apporte la mesure est moins élevée que ces coûts, on peut se questionner sur la nécessité d'utiliser la mesure en génie logiciel. Les coûts ne doivent pas être démesurés. L'incertitude est donc que la valeur apportée par les mesures n'est peut-être pas supérieure aux coûts qu'elle entraîne. Autrement dit, les avantages qu'apportent le fait de mesurer certains attributs ne sont pas nécessairement supérieurs aux coûts, soit parce que ces mesures n'améliorent pas réellement le processus ou les produits, soit parce que les coûts de collecte et d'analyse de l'information sont tout simplement trop élevés. Un exemple de ce genre de réflexion se trouve dans un article de Weller (1994) qui se pose la question suivante : combien de temps faut-il passer à mesurer les défauts d'un logiciel en développement? C'est une question très pertinente car la correction de défauts peut entraîner l'ajout d'autres défauts, créant ainsi un cercle vicieux. Alors, il est nécessaire de déterminer les critères d'acceptabilité afin d'avoir un logiciel de qualité suffisante sans trop dépenser à mesurer des défauts. Il faut penser également à la manière d'évaluer les bénéfices de la mesure versus ses coûts. Peut-être n'existe-t-il pas de méthodes pour les mesurer ou que leur validité peut être remise en question.

- La valeur apportée par les mesures n'est peut-être pas supérieure aux coûts qu'elle entraîne.
- Existe-t-il des méthodes permettant d'évaluer les bénéfices de la mesure versus ses coûts?
- Si ces méthodes existent, sont-elles valides?

5.2 MESURE POUR LA COMPRÉHENSION

Dans leur livre Software Metrics, Fenton et Pfleeger (1997) nous disent qu' « il y a des mesures qui nous aident à comprendre ce qui se passe durant le développement et la maintenance d'un logiciel. Nous évaluons la situation courante en établissant des points de repères (baselines) qui nous aident à établir des objectifs pour le comportement futur. En ce sens, les mesures rendent certains aspects des processus et des produits plus visibles à nos yeux, nous donnant ainsi une meilleure compréhension des liens entre les activités et les entités qu'elles affectent ».

Parmi les commentaires émits par les participants au cours des études Delphi, un seul se rattachait à la compréhension. Devons-nous penser que c'est parce qu'il est clair pour tout le monde que les mesures aident à la compréhension? Le paragraphe plus haut nous porte à croire que ce serait le cas. Cependant, il faut aussi envisager que c'est peut-être au contraire parce que l'on pense que les mesures n'aident pas en ce sens? Ce sont là des questions intéressantes qui peuvent également faire partie d'un questionnaire sur la place de la mesure en génie logiciel, même si elles ne sont pas rattachées à un commentaire des études Delphi.

Apport de la mesure à une meilleure compréhension du génie logiciel

Commentaire(s) : 35

L'unique commentaire concernant la mesure pour la compréhension affirme que la nécessité de mesurer est un signe que nous ne comprenons pas les problème/artefact/technologie. Le participant a aussi ajouté que la mesure peut être utile dans de telles situations. Ce qui est incertain ici c'est notre niveau de compréhension de ces problèmes/artefacts/technologies et également de savoir jusqu'à quel point la mesure peut aider à avoir une meilleure compréhension de ces choses. Une autre partie du commentaire nous fait penser également à d'autres incertitudes : quels aspects de ces entités est-il important de mesurer et quel poids faut-il donner à chacun de ces aspects? D'autre part, dans l'article « Using Metrics in Management Decision Making », Stark et Durst (1994) disent que prendre des mesures sur un projet rend les problèmes visibles. Cette meilleure visibilité des problèmes doit, selon nous, entraîner une meilleure compréhension. Cependant, cela demeure une

incertitude. Nous posons alors la question suivante : Le fait que l'utilisation des mesures dans un projet rende les problèmes plus visibles augmente-t-il la compréhension en génie logiciel?

- Quel est notre niveau de compréhension des problèmes/artefacts/technologies du génie logiciel?
- Jusqu'à quel point la mesure peut-elle aider à avoir une meilleure compréhension de ces choses?
- Quels aspects de ces entités est-il important de mesurer?
- Quel poids faut-il donner à chacun de ces aspects?
- Puisque les mesures sont sensées rendre les problèmes plus visibles, augmentent-elles, par le fait même, la compréhension des problèmes/artefacts/technologies en génie logiciel?

5.3 MESURE POUR L'ÉVALUATION

La mesure pour l'évaluation est très semblable à la mesure pour faire des prévisions car toutes deux nous permettent de mesurer les attributs des entités durant le processus de développement de logiciel. Cependant, la mesure pour l'évaluation se différencie de la mesure pour les prédictions en se concentrant sur le passé et sur le présent plutôt que sur l'avenir.

Les commentaires se rattachant à la mesure pour l'évaluation ne sont pas très nombreux. Nous en avons distingué de deux sortes : ceux concernant l'évaluation en général et d'autres de l'utilisation de l'évaluation pour diminuer l'incertitude des projets de génie logiciel. Nous traitons de ces deux groupes de commentaires dans les paragraphes qui suivent.

Qu'en est-il des mesures pour l'évaluation?

Commentaire(s) : 1, 7, 33

Les trois premiers commentaires concernant l'évaluation parlent de la nécessité d'établir des objectifs au départ et d'utiliser des mesures quantitatives afin d'évaluer la capacité d'atteinte de ces objectifs. Un des participants mentionne aussi qu'il faut disposer de telles mesures afin d'effectuer l'évaluation. Ceci nous amène à découvrir deux incertitudes de la mesure en génie logiciel. D'abord, nous nous demandons s'il existe des mesures adéquates permettant de réaliser l'évaluation en génie logiciel. Puis, nous voudrions savoir si ces mesures apportent réellement des améliorations au développement de logiciels.

- Existe-t-il des mesures adéquates, permettant de réaliser l'évaluation en génie logiciel?
- Ces mesures apportent-elles réellement des améliorations au développement de logiciels?

L'incertitude et la mesure pour l'évaluation

Commentaire(s) : 48, 68

Comme nous l'avons dit plus haut, d'autres commentaires se rapportent à l'incertitude dans les projets de développement de logiciel. Un participant a déclaré qu'à son avis l'incertitude est plus

grande en génie logiciel que dans n'importe quelle autre discipline du génie. Selon lui, les estimés d'effort et de coût des projets sont erronés parce qu'ils contiennent beaucoup d'incertitudes. Nous savons que bien des projets dépassent largement le temps et le budget qui leur avaient été alloués. Cependant, un autre participant a affirmé que l'incertitude existe en génie logiciel comme partout dans le monde. Selon lui, cette incertitude pourrait être diminuée si l'on se concentrait plus sur certaines tâches (telles que l'établissement de la faisabilité, la conception d'une architecture de base, l'identification et la réduction des risques, la construction d'un plan et l'estimation du temps et du budget) plutôt que d'embarquer immédiatement dans la construction des logiciels. Ces commentaires nous font penser à plusieurs incertitudes. Premièrement, la discipline du génie logiciel possède-t-elle un plus haut niveau d'incertitude que les autres disciplines du génie ou est-ce simplement que ces autres disciplines sont arrivées au fur et à mesure à maîtriser ces incertitudes? L'utilisation de la mesure dans les activités mentionnées plus haut ne pourrait-elle pas réellement aider à diminuer le niveau d'incertitudes des projets de génie logiciel?

- La discipline du génie logiciel possède-t-elle un plus haut niveau d'incertitude que les autres disciplines du génie ou est-ce simplement que ces autres disciplines sont arrivées au fur et à mesure à maîtriser ces incertitudes?
- L'utilisation de la mesure dans certaines activités (l'établissement de la faisabilité, la conception d'une architecture de base, l'identification et la réduction des risques, la construction d'un plan et l'évaluation du temps et du budget) ne pourrait-elle pas réellement aider à diminuer le niveau d'incertitudes des projets de génie logiciel?

L'incertitude dans l'évaluation

Commentaire(s) : 13 et 68

Nous avons parlé dans le paragraphe précédent de l'incertitude dans le génie logiciel. Cependant, des participants affirment qu'il y a aussi beaucoup d'incertitudes dans les estimés eux-mêmes. L'un d'entre eux a même déclaré qu'aucun estimé n'est exact, à moins que ce ne soit par chance. Il a donc proposé de prendre pour acquis que tout estimé est incorrect, plutôt sous-estimé que surestimé et a ajouté qu'il fallait mettre de côté les contingences. Pourtant, si le participant dit vrai, est-il réellement nécessaire de faire des estimés puisque nous pouvons nous retrouver avec un grand nombre de données erronées? Peut-être aussi que, malgré leur inexactitude, ces données nous donnent une approximation suffisante de ce que l'on veut évaluer, ce qui justifierait la nécessité de faire des estimés? Avant de nous poser ces questions, il faut également savoir s'il n'y a pas un

manque de confiance envers les estimés. Nous nous demandons également si un manque de confiance ne peut pas influencer les décisions, afin que seules l'expérience ou l'intuition soient utilisées.

- Y a-t-il un manque de confiance envers les estimés?
- Un manque de confiance envers les estimés ne peut-il pas influencer les prises de décision afin que celles-ci soient basées principalement sur l'expérience ou l'intuition?
- Puisque les estimés sont souvent incorrects, est-ce qu'il est réellement nécessaire d'en faire, puisqu'un grand nombre de données sont erronées?
- Les estimés nous donnent-ils, malgré leur inexactitude, une approximation suffisante de ce que l'on veut évaluer, ce qui justifierait la nécessité de faire des estimés?

5.4 MESURE POUR L'EXPÉRIMENTATION

Dans leur article «Experimentation in Software Engineering», Basili, Selby et Hutchens (1986) présentent l'expérimentation comme un moyen de mieux évaluer, prédire, comprendre, contrôler ainsi qu'améliorer le processus de développement de logiciels et le produit qui en résulte. En fait, les activités énumérées ci-dessus correspondent aux catégories du pourquoi mesurer sur lesquelles est basée notre recherche d'incertitudes. Cela signifie probablement que la mesure pour l'expérimentation rend plus facile le travail de mesure pour les autres catégories.

Tout comme dans les autres domaines, l'expérimentation en génie logiciel se fait en itération en émettant d'abord une hypothèse. Cette dernière est ensuite testée afin qu'elle soit raffinée ou remplacée. Basili, Selby et Hutchens (1986) nous disent que cette approche prend une importance spéciale en génie logiciel parce que nous avons grandement besoin d'améliorer nos connaissances concernant la manière dont les logiciels sont développés, les effets des diverses technologies et les aspects nécessitant le plus d'amélioration. De plus, ils ajoutent qu'il y a beaucoup à apprendre et que l'intuition n'est pas toujours le meilleur des professeurs.

Nous avons été surpris de ne trouver aucun commentaire concernant la mesure pour l'expérimentation en génie logiciel, premièrement parce que, comme nous l'avons dit plus haut, l'expérimentation serait favorable au génie logiciel et, deuxièmement, parce que la mesure occupe habituellement une place importante dans les processus d'expérimentation. Nous nous sommes dit que cette absence de commentaires portant sur la mesure pour l'expérimentation était peut-être due au fait qu'on n'expérimente pas assez en génie logiciel. Plus particulièrement, on ne fait pas assez d'expérimentation rigoureuse et quantitative. En effet, on se contente plutôt de faire des analyses théoriques qui ne sont pas appliquées par la suite à la pratique (Fenton et Pfleeger, 1994).

Cependant, en cherchant plus d'information sur l'expérimentation dans le domaine informatique, nous avons découvert que les gens ne s'entendaient pas sur la nécessité d'y faire plus d'expérimentation. Nous avons consulté deux articles couvrant l'expérimentation en général, incluant le génie logiciel. L'une des raisons données par l'auteur est que ce qu'on y fait ressemble plus à du génie qu'à de la science (Tichy, 1998). D'autre part, les membres du domaine informatique semblent se faire confiance entre eux et ne pas vouloir mettre à l'épreuve ce que font les autres, d'où

un problème de manque de validation pratique des théories proposées. Zelkowitz et Wallace (1998) ont étudié les méthodes d'expérimentation de 600 études et ont conclu qu'un trop grand nombre de ces études n'avait pas de validations expérimentales ou que celles-ci étaient informelles. Ceci nous donne deux incertitudes concernant la mesure pour l'expérimentation en génie logiciel. Faut-il expérimenter en génie logiciel, bien que ce ne soit pas une science? En d'autres termes, si ce doit être du génie, faut-il expérimenter? L'expérimentation apportera-t-elle beaucoup au génie logiciel en général? Est-ce que des organisations en particulier pourraient trouver des bénéfices à pratiquer des expérimentations en génie logiciel? Dans les cas d'expérimentation en génie logiciel, y a-t-il un manque de validation formelle?

- Faut-il expérimenter en génie logiciel, bien que ce ne soit pas une science?
- Si oui, est-ce que des organisations en particulier pourraient trouver des bénéfices à pratiquer des expérimentations en génie logiciel?
- Dans les cas d'expérimentation en génie logiciel, y a-t-il un manque de validation formelle?

5.5 MESURE POUR LE CONTRÔLE DE PROJET

Fenton et Pfleeger (1997) décrivent l'utilisation de la mesure dans le contrôle de projet de la façon suivante : « La mesure nous permet de contrôler ce qui se passe dans nos projets. En utilisant nos points de repères, nos objectifs et notre compréhension des liens, nous prédisons ce qui arrivera vraisemblablement et nous modifions les processus et les produits afin d'arriver à atteindre nos objectifs.

Utilité de la mesure dans le contrôle de projet

Commentaire(s) : 7 et 9

Chaque chef de projet devrait planifier et spécifier, de manière quantitative, ses stratégies de fiabilité en se basant sur ses objectifs de fiabilité. C'est ce que dit l'un des participants alors qu'un autre participant a affirmé qu'il faudrait vérifier le niveau de fiabilité à différentes étapes du projet et comparer les résultats obtenus avec les objectifs fixés au départ. Il est courant d'entendre que plus les objectifs d'un projet sont bien définis, plus petits sont les risques d'échec. D'ailleurs, le principe fondamental le plus populaire dans nos études n'est-il pas celui qui demande d'investir plus de temps dans la compréhension du problème? Cela a pour but, entre autres, de pouvoir mieux planifier et contrôler le projet. Cependant, la question que nous nous posons est de savoir s'il est réellement utile de se baser sur des mesures quantitatives afin d'effectuer ce contrôle et cette planification. Est-ce que cette activité n'implique pas automatiquement la nécessité de mesures quelconques? Nous pouvons ainsi définir une autre incertitude : est-ce que tout contrôle de projets en développement de logiciels implique la nécessité d'utiliser des mesures quantitatives? Weller (1994), dans son article « Using Metrics to Manage Software Projects » affirme que tous les projets, sauf les plus petits, requièrent des mesures pour un gestion efficace du projet. Fenton et Pfleeger (1997) citent DeMarco, qui dit qu' « on ne peut contrôler ce qu'on ne peut mesurer ». Fenton et Pfleeger parlent aussi de l'approche GQM (Goal – Question – Metric) de Basili et de ses collègues. Cette approche a pour but d'aligner la prise de mesures dans les différents projets avec les objectifs généraux de l'entreprise qui effectuent ces mesures. AT&T, par exemple, a utilisé GQM pour aider à déterminer les mesures appropriées afin d'évaluer leur processus d'inspection. Leurs buts étaient les suivant : planifier, faire le suivi et contrôler ainsi qu'améliorer leur processus d'inspection

(Fenton et Pfleeger, 1997). Une telle approche contribue probablement à mieux gérer les projets en fonction des besoins d'une entreprise. Cette approche n'est cependant pas parfaite. Fenton et Pfleeger mentionnent, entre autres, la nécessité d'avoir un modèle combinant les mesures, ceci dans le but de pouvoir répondre aux questions posées.

- Est-il réellement utile de se baser sur des mesures quantitatives afin d'effectuer le contrôle et la planification de projets?
- Est-ce que le contrôle n'implique pas automatiquement la nécessité de mesures quelconques?
- Est-ce que tout contrôle de projets en développement de logiciels implique la nécessité d'utiliser des mesures quantitatives?

Utilité de la mesure dans le choix d'outils et de méthodes pour le contrôle de projet

Commentaire(s) : 10, 83

Un commentaire dit que les projets devraient choisir leurs outils et méthodes en se basant sur des critères quantitatifs, en mesurant leur effet sur la fiabilité, l'échéancier et les coûts par rapport aux coûts d'implantation. Par contre, un autre participant disait que le choix de ces outils et méthodologies pourrait plutôt être basé sur une analyse du marché. Tous n'approuvent donc pas le fait d'effectuer ces choix selon des critères quantitatifs. Nous sommes alors incertains de la nécessité d'utiliser ces critères quantitatifs dans le choix des outils et des méthodes. Un danger que nous avons identifié lors de la revue de la littérature (Fenton et Pfleeger, 1994) est de croire ce que disent les fabricants de certains outils. Ces derniers prétendent souvent que l'utilisation de leurs outils et méthodes peuvent améliorer la productivité ou réduire les efforts au point de vue de la maintenance. Ils donnent même parfois des pourcentages pour donner plus de poids à leur argument. Cependant, la manière de calculer ces pourcentages d'amélioration n'est souvent pas mentionnée, n'étant probablement pas basée sur de réelles mesures.

- Est-il nécessaire d'utiliser des critères quantitatifs dans le choix des outils et méthodes de projets?

La mesure pour évaluer le niveau de risque d'un projet

Commentaire(s) : 11

Le commentaire dont nous traitons ici parle de risques dans un projet. Selon nous, pour avoir le contrôle d'un projet, il faut diminuer au maximum les risques d'échec de ce projet. C'est pourquoi, nous pensons que c'est un sujet intéressant à traiter au sein du thème de mesure pour le contrôle de projet. D'après le commentaire, il n'est pas possible d'éliminer les risques dans un projet, mais on peut les documenter, planifier et concevoir en fonction de ces risques et aussi accepter ces derniers et les mesurer, afin de les réduire à des niveaux acceptables. Nous retenons ici le mot «mesurer » et nous nous demandons s'il est réellement nécessaire de mesurer ces risques afin de savoir s'ils sont acceptables? Par ailleurs, une étape importante dans la gestion de projets est l'analyse de risques effectuées avant le début du projet. Cette analyse consiste à déterminer si les risques d'échec sont suffisamment bas pour permettre de réaliser le projet. Il nous semble que des données quantitatives pourraient aider à prendre une telle décision. Cependant, cela demeure une incertitude pour nous. Dans son article « Software Risk Management : Principles and Practices », Boehm (1991) dit que les études postmortem des projets informatiques qui se sont révélés être des désastres indiquaient que les problèmes auraient pu être évités s'il y avait eu, tôt dans le projet, un souci d'identifier et de résoudre les éléments à haut risque. Boehm dit également que le degré d'incertitude est aussi une source majeure de risque et que la meilleure manière de le réduire est de se procurer des informations sur la situation actuelle. Nous pensons que l'un des meilleurs moyens de se procurer ces informations est probablement de prendre des mesures de cette situation.

- Est-il absolument nécessaire de mesurer les risques de projets afin de pouvoir déterminer s'ils sont acceptables ou non?
- Les données quantitatives peuvent-elles aider à déterminer avant le début d'un projet si celui-ci présente de trop grands risques?

La mesure pour le contrôle de la qualité et des coûts

Commentaire(s) : 3, 12

Certains participants ont mentionné la difficulté de contrôler la qualité et les coûts. À leur opinion, pour en avoir le contrôle, il faut absolument définir des exigences mesurables. Pour cette raison, l'utilisation de mesures quantitatives y serait essentielle. La question que nous nous posons est la suivante : faut-il réellement avoir des mesures quantitatives afin de vérifier l'atteinte des exigences de coûts et de qualité définies au préalable?

- Faut-il réellement avoir des mesures quantitatives afin de vérifier l'atteinte des exigences de coûts et de qualité définies au préalable?

Limites de l'utilisation de la mesure dans la prise de décision

Commentaire(s) : 15

L'un des participants a suggéré de préciser le principe A en le formulant de la manière suivante : Appliquer et utiliser des mesures quantitatives dans la prise de décision pour la gestion de projet et pour le développement technique. Faut-il réellement limiter l'utilisation de mesures quantitatives à ces deux activités ou bien faut-il le faire de manière générale dans toutes prises de décision? C'est cette incertitude que nous évoquons ici.

- Faut-il réellement limiter l'utilisation de mesures quantitatives à la gestion de projet et au développement technique ou bien faut-il le faire de manière générale dans toutes prises de décision?

Profil d'utilisation

Commentaire(s): 8

Dans le commentaire traité dans le présent paragraphe, le participant parle de développer un ou plusieurs profils d'utilisation pour chaque projet, en consultation avec les utilisateurs ou leurs représentants, ceci, dans le but de pouvoir quantifier le taux d'utilisation auquel on s'attend et aussi de mettre l'accent sur les fonctions les plus critiques ou les plus utilisées, lors du développement et des tests. En étudiant ce commentaire, nous nous posons les questions suivantes : Existe-t-il de bonnes mesures permettant de quantifier le taux d'utilisation d'un système? Puisqu'on parle de mettre l'accent sur les fonctions les plus critiques ou les plus utilisées, existe-t-il des mesures permettant de les détermine?

- Existe-t-il de bonnes mesures permettant de quantifier le taux d'utilisation d'un système?
- Puisqu'on parle de mettre l'accent sur les fonctions les plus critiques ou les plus utilisées, existe-t-il des mesures permettant de déterminer quelles fonctions sont les plus critiques et les plus utilisées?

5.6 MESURE POUR L'AMÉLIORATION DU PROCESSUS

La qualité du processus utilisé dans le développement de logiciels nous semble être un élément important en génie logiciel puisqu'elle a des effets sur la qualité des logiciels ainsi que sur le temps et les coûts de production. La mesure nous permet d'améliorer la qualité des processus de développement de logiciels en nous permettant de mieux les évaluer et de mieux comprendre la portée des changements qui y sont apportés. C'est d'ailleurs pourquoi plusieurs instruments d'évaluation de ces processus existent, dont le plus connu est le modèle CMM du Software Engineering Institute (SEI, 1995). Certains commentaires des études Delphi parlent justement de l'utilisation de la mesure dans le but d'améliorer les processus de développement du logiciel. Ces commentaires sont traités dans les paragraphes qui suivent.

Importance du rôle de la mesure dans l'amélioration du processus

Commentaire(s) : 5, 80, 83

Selon le commentaire d'un participant, il faudrait un cycle de production orienté vers l'amélioration, pour le développement de logiciels. D'après ce participant, cette tâche est complexe et devrait être gérée de manière plus contrôlée. De plus, il pense que l'utilisation des mesures de processus et de produits devrait constituer un composant intégral de ce processus de contrôle. En nous basant sur nos différentes références et sur les commentaires des participants, nous pouvons dire que la mesure joue un rôle important dans l'amélioration du processus. Dans la revue de littérature également, la mesure est présentée comme un outil pouvant aider à améliorer le processus. Daskalantonakis nous le présente bien dans son article «Achieving Higher SEI Levels » (1994). En effet, il y cite l'approche de Motorola face aux mesures de logiciels : « Mesurer n'est pas le but. Le but est l'amélioration (du processus) par la mesure, l'analyse et le feedback. Ce qui nous paraît cependant plus incertain, c'est le degré d'importance de la mesure par rapport aux autres facteurs d'amélioration du processus. Autrement dit, y a-t-il d'autres facteurs plus efficaces pour l'amélioration du processus que le fait de le mesurer? Fenton et Pfleeger (1997) pensent que le lien entre la maturité des processus et la prise de mesures est très fort. Ils disent aussi que plus les processus sont matures, plus la prise de mesures est mature. Cependant, si on se fie au modèle CMM mentionné plus haut, il semblerait que cela va de soi et que l'utilisation de mesures pour gérer

le processus de développement de logiciel implique obligatoirement une amélioration de ce processus. Nous pensons malgré tout que cela n'est pas si évident et que le lien entre les mesures utilisées pour gérer le processus et l'amélioration du niveau de maturité de ce processus demeure une incertitude en génie logiciel.

- Y a-t-il d'autres facteurs permettant beaucoup plus l'amélioration du processus que le fait de le mesurer?
- Le lien entre les mesures utilisées pour gérer le processus de développement de logiciels et l'amélioration du niveau de maturité de ce processus est-il évident?

Mesure de la qualité du processus

Commentaire(s) : 65

Un des participants a fait remarquer qu'il était nécessaire de quantifier le niveau de qualité du processus et du produit lors de la quantification des exigences de performance du produit. Nous rappelons qu'une question sur la nature de la qualité avait déjà été posée. Il est vrai que chacun peut considérer la qualité selon des aspects différents en l'absence de normes énonçant clairement la définition de la qualité en génie logiciel. Cependant, nous pensons qu'il est essentiel de poursuivre les analyses. L'incertitude que nous trouvons ici est de savoir s'il est possible de mesurer la qualité alors que celle-ci n'est pas bien définie. Nous nous demandons également s'il vaut mieux ne rien mesurer du tout plutôt que de mesurer quelque chose, notamment la qualité, en fonction de la perception que nous en avons?

Est-il possible de mesurer la qualité alors que celle-ci n'est pas bien définie?
Est-il mieux de ne rien mesurer du tout plutôt que de mesurer quelque chose, notamment la qualité, en fonction de la perception que nous en avons?

Validité des mesures utilisées dans le but d'améliorer le processus

Commentaires : 32, 49

Les prochains commentaires que nous analyserons parlent de manière plus concrète de l'amélioration du processus de par l'utilisation de mesures. En effet, un participant a suggéré qu'il

faudrait utiliser ces données quantitatives non seulement pour guider la gestion des projets mais aussi pour contrôler (gauge) les progrès dans l'amélioration du processus. Ils disent, de plus, que le fait de mesurer les progrès et la performance des processus de développement de logiciels est essentiel aux améliorations nécessaires pour satisfaire la demande de l'industrie. Les commentaires précédents ayant souvent remis en question la validité des mesures en génie logiciel, cet aspect nous semblent donc incertain. Et puisque beaucoup semblent être d'accord sur l'importance du rôle de la mesure dans l'amélioration du processus, nous nous demandons s'il existe également des mesures permettant de contrôler l'amélioration apportée aux processus grâce à l'utilisation de certaines mesures.

- Les mesures permettant de mesurer l'amélioration des processus en génie logiciel sont-elles valides?
- Existe-t-il des mesures permettant de contrôler l'amélioration apportée aux processus grâce à l'utilisation de certaines mesures?

La mesure permet-elle au processus de développement de logiciels de ressembler plus aux processus utilisés dans les autres disciplines du génie?

Commentaire(s) : 17, 67

En parlant du principe A qui, nous le rappelons, conseille d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision, l'un des participants a affirmé que c'est l'un des seuls mécanismes qui puissent nous permettre d'améliorer les processus afin de les rendre semblables à ceux des autres disciplines du génie. Un autre participant a ajouté que la formation des ingénieurs en logiciel devrait comprendre l'enseignement des fondements scientifiques et d'ingénierie. Ceci rejoint la discussion que nous avons eu plus haut, à savoir que sans l'utilisation de la mesure, nous n'avons pas de réelle discipline du génie. Nous nous demandons alors si on peut parler de discipline du génie dans une branche où la mesure ne joue pas un rôle important? Nous demandons aussi si l'accent est suffisamment mis sur la mesure dans les programmes de formation en génie logiciel. Une autre incertitude plus directement liée aux commentaires auxquels se réfère ce paragraphe, est de savoir si la mesure constitue réellement l'un des seuls mécanismes permettant l'amélioration du processus.

- Est-ce qu'on peut parler de discipline du génie dans une branche où la mesure ne joue pas un rôle important?
- L'accent est-il mis suffisamment sur la mesure dans les programmes de formation en génie logiciel?
- La mesure constitue-t-elle réellement l'un des seuls mécanismes permettant l'amélioration du processus?

5.7 MESURE POUR L'AMÉLIORATION DU PRODUIT

Le produit est le but final du processus de développement de logiciel. Et dans la mesure du possible, nous devons tenter de créer un produit répondant aux besoins réels des clients. C'est pourquoi, il est important de chercher à améliorer le produit. La mesure peut jouer un rôle important à ce niveau en vérifiant, d'une part, le niveau de qualité du produit, d'autre part, les améliorations enregistrées dans ce dernier. Dans cette section, nous traitons des commentaires portant sur l'utilisation de la mesure dans le processus d'amélioration du produit.

Certains commentaires traitent à la fois de l'amélioration du processus et de celle du produit. Ils ont déjà été traités dans la partie sur la mesure pour l'amélioration du processus. Nous ne les reprendrons pas dans la présente section. Ce sont, notamment, les commentaires 65 et 67.

Les bénéfices de l'utilisation de la mesure dans l'amélioration du produit

Commentaire(s) : 4

Le premier commentaire concernant ce thème affirme que l'évaluation du niveau de qualité d'un logiciel doit être scientifique et quantitative. Cependant, les résultats doivent être utiles et le coût ne doit pas dépasser les avantages. Nous avions déjà discuté de cet aspect dans la partie sur la mesure en général. Nous essaierons donc d'être plus spécifique dans la détermination de l'incertitude contenue dans ce commentaire. Nous nous demandons s'il faut à tout prix viser à utiliser des mesures permettant l'amélioration du produit, ou s'il est important de calculer le retour sur les investissements que l'on tire à utiliser ces mesures?

- Faut-il à tout prix viser à utiliser des mesures permettant l'amélioration du produit, ou est-il important de calculer le retour sur les investissements que l'on tire à utiliser ces mesures?

Validité des mesures pour l'amélioration du produit

Commentaire(s) : 47

Le contenu du prochain commentaire que nous analysons est à peu près le même. Il se rapporte au principe N qui dit d'améliorer la conception en étudiant les solutions précédentes à des problèmes similaires. Cette fois-ci, le participant dit que c'est une bonne chose d'utiliser ces mesures si le matériel est présent et que le temps d'étude n'est pas trop long,. Dans le cas contraire, il faudrait calculer le retour sur l'investissement qu'apporte l'utilisation de ces mesures. Nous savons qu'il existe des mesures permettant de calculer l'amélioration du produit. Cependant, nous nous posons les questions suivantes : Ces mesures sont-elles valides? Autrement dit, permettent-elles réellement d'apporter une amélioration aux produits? De plus, tout comme nous nous sommes posé la question sur l'amélioration du processus, nous nous sommes également demandé s'il existe des mesures permettant d'évaluer l'amélioration d'un produit, due à l'utilisation de mesures? D'autre part, on pourrait se demander si l'utilisation de ces mesures apporte réellement un bénéfice?

- Les mesures permettant l'amélioration du produit sont-elles valides?
- Existe-t-il des mesures permettant d'évaluer l'amélioration d'un produit, due à l'utilisation de mesures?
- L'utilisation de telles mesures apporte-t-elle réellement un bénéfice?

Les compromis d'un produit d'ingénierie

Commentaire(s) : 66

Nous avons débuté cette section en mentionnant que le produit final est le but ultime de tout le processus de développement de logiciel et qu'il est donc important d'avoir un produit répondant aux réels besoins du client. Cependant, comme l'a fait remarquer un participant, il faut toujours faire des compromis. Le participant a précisé que les compromis de tous produits d'ingénierie doivent considérer les relations existant entre les coûts, l'échéancier, la performance technique du produit et la qualité des processus et du produit final. Jusqu'à présent nous avons parlé de mesures qui se consacraient à une tâche. Par contre, ces mesures devraient permettre de percevoir l'effet de la modification d'une composante sur les autres attributs. Ou encore, il faudrait combiner plusieurs mesures pour arriver à évaluer les bénéfices de ces compromis. Les incertitudes que nous

découvrons ici sont les suivantes : Existe-t-il des mesures permettant d'évaluer tous les aspects mentionnés par le participant, ceci dans le but de vérifier les bénéfices apportés par les compromis? Sinon, une combinaison de plusieurs mesures, se concentrant chacune sur un aspect en particulier, peut-elle permettre d'évaluer de tels bénéfices?

- Existe-t-il des mesures permettant d'évaluer les relations entre les coûts, l'échéancier, la performance technique du produit, et la qualité des processus et du produit final, ceci dans le but de vérifier les bénéfices apportés par les compromis entre ces différents aspects?
- Sinon, une combinaison de plusieurs mesures, se concentrant chacune sur un aspect en particulier, peut-elle permettre d'évaluer de tels bénéfices?

5.8 MESURE POUR LES PRÉDICTIONS

Comme nous l'avons mentionné plus haut, la mesure pour l'évaluation et celle pour les prévisions sont très semblables en ce sens que tous deux nous permettent de mesurer les attributs des entités tout au long du processus de développement de logiciel. Cependant, la mesure pour les prédictions se concentre sur l'avenir tandis que la mesure pour l'évaluation porte son attention sur le passé et le présent.

La mesure pour les prédictions dépend cependant de la mesure pour l'évaluation. En effet, il est nécessaire de pouvoir prédire les attributs déterminants dans le succès des projets. Pour ce faire, il faut se baser sur des mesures de bonne qualité prises pour des projets déjà terminés plutôt que de se fier uniquement à ses intuitions.

Pour certains projets, il est important de pouvoir estimer à l'avance leur coût et leur durée car il est simplement inacceptable de l'apprendre à la fin du projet. Souvent, il est aussi nécessaire de savoir si le produit pourra atteindre les objectifs de qualité désirés, en matière de fiabilité par exemple. Pour ce genre de projet, il est primordial de pouvoir prendre des mesures afin de prédire ce qui se passera si le projet est réalisé. En génie logiciel, des projets de ce genre existent. C'est pourquoi, la mesure pour les prédictions y tient une place importante. Cette importance est d'ailleurs mise en relief par Fenton et Pfleeger (1997) qui suggèrent de remplacer les décisions prises au hasard par un système de prédiction quantitatif.

Nous avons été surpris de ne trouver qu'un commentaire concernant les prédictions. Cependant, il ne faut pas oublier que notre source de données ne concernait pas uniquement la mesure et que de ce fait, nous n'y retrouverons pas nécessairement tous les aspects de la mesure.

Mesures étalons

Commentaire(s) : 1

L'unique commentaire concernant la mesure dit de prédire ce à quoi on s'attend des mesures avant de les prendre et d'utiliser les résultats afin de guider les ajustements. Dans les prédictions, il y a le

danger d'être obligé de mesurer quelque chose que nous ne connaissons généralement qu'imparfaitement. Il est donc plus facile de se tromper. Pourtant, lorsque le participant affirme que les résultats doivent guider les ajustements, nous sommes portés à nous demander si ce sont les résultats attendus qu'il faut établir comme étant exacts, ou si ce sont les résultats réels. En fait, nous pensons que les deux types de résultats peuvent être erronés. Nous sommes ainsi portés à nous demander à quoi il faut le plus se fier dans les cas de prédictions : aux mesures prévues au départ ou aux mesures prélevées en cours de projet. Cependant, cela nous amène à parler d'une autre possibilité, celle d'utiliser des données historiques afin d'établir les prédictions pour un projet futur. Dans ce cas, il nous semble que les résultats attendus auraient plus de poids. Il existe, par exemple, une base de donnée publique appelée ISBSG (référence). Elle contient toute une série d'informations sur des projets passés et permet de consulter des données sur des projets qui auraient des caractéristiques similaires au projet que l'on désire entreprendre. À notre avis, des mesures basés sur des données de projets déjà réalisés sont plus robustes que celles proposées par le participant dans son commentaire. Cependant cela demeure une incertitude. Alors, nous nous posons la question suivante : Est-ce que l'utilisation de mesures prises pour des projets antérieurs peut permettre de faire de meilleures prédictions sur des projets à venir?

- Dans les cas de prédictions, à quoi faut-il le plus se fier : aux mesures prévues au départ ou aux mesures prélevées en cours de projet?
- Est-ce que l'utilisation de mesures prises pour des projets antérieurs peut permettre de faire de meilleures prédictions sur des projets à venir?

6. SYNTHÈSE DE LA RECHERCHE

6.1 LISTE D'INCERTITUDES SUR LA PLACE DE LA MESURE EN GÉNIE LOGICIEL

La liste des questions qui suit constitue le livrable de la recherche : le questionnaire sur la place qu'occupe la mesure en génie logiciel. Nous rappelons que ce questionnaire n'est pas quelque chose de prêt à être soumis à un ensemble de participants. Il constitue plutôt un point de départ pour des investigations plus profondes sur la place qu'occupe la mesure en génie logiciel. L'idée de départ est que si réellement le génie logiciel est une branche du génie, la mesure doit y tenir une place importante.

Les questions que nous posons sont classées selon les catégories de raisons de mesurer identifiées au début de la recherche. Les questions présentées dans ce chapitre sont parfois légèrement différentes de celles exposées dans le chapitre sur l'analyse de la place de la mesure. À notre avis, une reformulation de certaines questions contribuait à une meilleure compréhension des incertitudes hors de leur contexte de discussion.

6.1.1 Incertitudes liées à la mesure en général

- La mesure joue-t-elle un rôle primordial en génie logiciel?
- Si la mesure ne joue pas un rôle primordial en génie logiciel, les chances de cette discipline d'être reconnue par différents organismes comme branche du génie sont-elles automatiquement éliminées?
- Certains semblent dire qu'il n'est pas toujours pratique de prendre des mesures en génie logiciel. Cependant, est-ce une raison suffisante pour ne pas en prendre?
- Est-ce nécessaire de mesurer certaines entités en génie logiciel si nous ne connaissons pas très bien ces entités?
- Quels avantages nous apporte l'utilisation de mesures en génie logiciel?
- Quelle quantité d'information faut-il avoir sur un attribut avant de considérer qu'il est raisonnable de le mesurer?
- Comment savoir si on a réellement mesuré l'attribut à mesurer?
- Que faut-il mesurer en génie logiciel?
- Puisque l'intuition et les croyances seront probablement utilisées dans la prise de décision en génie logiciel, ne faut-il pas plutôt essayer de baser cette prise de décision sur des données quantitatives?
- Avons-nous une compréhension suffisante du domaine du génie logiciel pour nous permettre d'être quantitatif?

- Ne faut-il pas arriver à maîtriser des aspects de base du génie logiciel, telle que la compréhension du problème, avant de s'aventurer à utiliser les mesures quantitatives dans la prise de décision?
- Pour que les gestionnaires de génie logiciel arrivent à avoir des critères aussi bons que ceux utilisés dans la gestion des autres activités d'ingénierie, ne faut-il pas justement tenter de mesurer certains attributs des entités afin de mieux les connaître?
- Les mesures en génie logiciel sont-elles réellement immatures et non fiables?
- Le qualitatif a-t-il plus d'importance que le quantitatif dans la prise de décision?
- Quand, pourquoi et comment faut-il utiliser le quantitatif dans la prise de décision?
- Est-il essentiel d'appliquer et d'utiliser des mesures quantitatives dans la prise de décision?
- Peut-on mesurer la qualité en génie logiciel si le terme « qualité » n'y est pas bien défini?
- S'il faut établir des objectifs de qualité pour chaque produit livrable, faut-il le faire de manière à pouvoir les mesurer?
- Peut-on mesurer des objectifs de qualité pour chaque produit livrable?
- Est-ce que seul les organisations plus matures peuvent établir des objectifs de qualité?
- Selon un participant, il faut une organisation très mature pour établir des objectifs de qualité. Cependant, n'y a-t-il pas moyen pour des organisations moins matures d'arriver à se fixer des objectifs de qualité?
- Faut-il toujours utiliser des mesures quantitatives dans la prise de décision?
- S'il n'est pas nécessaire de toujours utiliser des mesures quantitatives dans la prise de décision, comment savoir quand appliquer ces mesures?
- L'utilisation de données uniquement non quantitatives peut-elle satisfaire pour la prise de décision ou faut-il y ajouter des mesures quantitatives?
- L'utilisation de mesures quantitatives peut-elle réellement permettre d'identifier les incertitudes d'un projet?
- Une fois les incertitudes d'un projet identifiées, les mesures peuvent-elles nous aider à mieux les gérer?
- La mesure pourrait-elle indirectement faciliter la prise de décision si elle permettait réellement de diminuer le niveau d'incertitude dans les projets?
- La mesure peut-elle nous aider à améliorer la justesse des estimés?
- Les outils et les méthodes de mesure utilisés en génie logiciel sont-ils valides?
- À quel point la validité interne des mesures en génie logiciel est-elle respectée?
- À quel point peut-on parler de validité externe des données en génie logiciel?
- Est-ce que des mesures inexactes peuvent être utiles en génie logiciel et, si oui, que doit être la marge d'erreur à ne pas dépasser?
- En utilisant les mesures, quelle affirmation sensée peut on faire à propos des attributs et des entités qui les possèdent?
- Quelles opérations sensées peut-on effectuer sur les mesures?
- La valeur apportée par les mesures est-elle supérieure aux coûts qu'elle entraîne?
- Existe-t-il des méthodes permettant d'évaluer les bénéfices de la mesure versus ses coûts?
- Si des méthodes permettant d'évaluer les bénéfices de la mesure versus ses coûts existent, sont-elles valides?

6.1.2 Incertitudes liées à la mesure pour la compréhension

- Quel est notre niveau de compréhension des problèmes/artefacts/technologies du génie logiciel?
- Jusqu'à quel point la mesure peut-elle aider à avoir une meilleure compréhension des problèmes/artefacts/technologies?
- Quels aspects de ces entités est-il important de mesurer?
- Quel poids faut-il donner à chacun de ces aspects?
- Puisque les mesures sont sensées rendre les problèmes plus visibles, augmentent-elles, par le fait même, la compréhension des problèmes/artefacts/technologies en génie logiciel?

6.1.3 Incertitudes liées à la mesure pour l'évaluation

- Existe-t-il des mesures adéquates, permettant de réaliser l'évaluation en génie logiciel?
- Les mesures pour l'évaluation apportent-elles réellement des améliorations au développement de logiciels?
- La discipline du génie logiciel possède-t-elle un plus haut niveau d'incertitude que les autres disciplines du génie ou est-ce simplement que ces autres disciplines sont arrivées au fur et à mesure à maîtriser ces incertitudes?
- L'utilisation de la mesure dans certaines activités (l'établissement de la faisabilité, la conception d'une architecture de base, l'identification et la réduction des risques, la construction d'un plan et l'évaluation du temps et du budget) ne pourrait-elle pas réellement aider à diminuer le niveau d'incertitudes des projets de génie logiciel?
- Y a-t-il un manque de confiance envers les estimés?
- Un manque de confiance envers les estimés ne peut-il pas influencer les prises de décision afin que celles-ci soient basées principalement sur l'expérience ou l'intuition?
- Puisque les estimés sont souvent incorrects, est-ce qu'il est réellement nécessaire d'en faire, puisqu'un grand nombre de données sont erronées?
- Les estimés nous donnent-ils, malgré leur inexactitude, une approximation suffisante de ce que l'on veut évaluer, ce qui justifierait la nécessité de faire ces estimés?

6.1.4 Incertitudes liées à la mesure pour l'expérimentation

- Faut-il expérimenter en génie logiciel, bien que ce ne soit pas une science?
- Si oui, est-ce que des organisations en particulier pourraient trouver des bénéfices à pratiquer des expérimentations en génie logiciel?
- Dans les cas d'expérimentation en génie logiciel, y a-t-il un manque de validation formelle?

6.1.5 Incertitudes liées à la mesure pour le contrôle de projet

- Est-il réellement utile de se baser sur des mesures quantitatives afin d'effectuer le contrôle et la planification de projets?
- Est-ce que le contrôle n'implique pas automatiquement la nécessité de mesures quelconques?

- Est-ce que tout contrôle de projets en développement de logiciels implique la nécessité d'utiliser des mesures quantitatives?
- Est-il nécessaire d'utiliser des critères quantitatifs dans le choix des outils et méthodes de projets?
- Est-il absolument nécessaire de mesurer les risques de projets afin de pouvoir déterminer s'ils sont acceptables ou non?
- Les données quantitatives peuvent-elles aider à déterminer avant le début d'un projet si celui-ci présente de trop grands risques?
- Faut-il réellement avoir des mesures quantitatives afin de vérifier l'atteinte des exigences de coûts et de qualité définies au préalable?
- Faut-il réellement limiter l'utilisation de mesures quantitatives à la gestion de projet et au développement technique ou bien faut-il le faire de manière générale dans toutes prises de décision?
- Existe-t-il de bonnes mesures permettant de quantifier le taux d'utilisation d'un système?
- Puisqu'on parle de mettre l'accent sur les fonctions les plus critiques ou les plus utilisées, existe-t-il des mesures permettant de déterminer quelles fonctions sont les plus critiques et les plus utilisées?

6.1.6 Incertitudes liées à la mesure pour l'amélioration du processus

- Y a-t-il d'autres facteurs permettant beaucoup plus l'amélioration du processus que le fait de le mesurer?
- Le lien entre les mesures utilisées pour gérer le processus de développement de logiciels et l'amélioration du niveau de maturité de ce processus est-il évident?
- Est-il possible de mesurer la qualité du processus alors que celle-ci n'est pas bien définie?
- Est-il mieux de ne rien mesurer du tout plutôt que de mesurer quelque chose, notamment la qualité, en fonction de la perception que nous en avons?
- Les mesures permettant de mesurer l'amélioration des processus en génie logiciel sont-elles valides?
- Existe-t-il des mesures permettant de contrôler l'amélioration apportée aux processus, amélioration due à l'utilisation d'autres mesures?
- Est-ce qu'on peut parler de discipline du génie dans une branche où la mesure ne joue pas un rôle important?
- L'accent est-il mis suffisamment sur la mesure dans les programmes de formation en génie logiciel?
- La mesure constitue-t-elle réellement l'un des seuls mécanismes permettant l'amélioration du processus. ?

6.1.7 Incertitudes liées à la mesure pour l'amélioration du produit

- Faut-il à tout prix viser l'utilisation de mesures permettant l'amélioration du produit, ou est-il important de calculer le retour sur les investissements que l'on tire à utiliser ces mesures?
- Les mesures permettant l'amélioration du produit sont-elles valides?

- Existe-t-il des mesures permettant d'évaluer l'amélioration d'un produit, due à l'utilisation de mesures?
- L'utilisation de mesures pour l'amélioration du produit apporte-t-elle réellement un bénéfice?
- Existe-t-il des mesures permettant d'évaluer les relations entre les coûts, l'échéancier, la performance technique du produit, et la qualité des processus et du produit final, ceci dans le but de vérifier les bénéfices apportés par les compromis entre ces différents aspects?
- Sinon, une combinaison de plusieurs mesures, se concentrant chacune sur un aspect en particulier, peut-elle permettre d'évaluer de tels bénéfices?

6.1.8 Incertitudes liées à la mesure pour les prédictions

- Dans les cas de prédictions, à quoi faut-il le plus se fier : aux mesures prévues au début d'un projet ou aux mesures prélevées en cours de projet?
- Est-ce que l'utilisation de mesures prises pour des projets antérieurs peut permettre de faire de meilleures prédictions sur des projets à venir?

6.2 RETOUR SUR LES OBJECTIFS ET APPORT DE LA RECHERCHE

Notre objectif principal, dans cette recherche, était 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. L'identification des incertitudes concernant la mesure en génie logiciel constituait un sous-objectif à la recherche.

Comme nous l'avions prévu, nous avons extrait des commentaires sur les principes fondamentaux une longue liste de points ambigüs ou imprécis concernant la mesure en génie logiciel. À partir de cette liste, nous avons produit le questionnaire sur la place de la mesure en génie logiciel. Ce questionnaire pourra servir à évaluer la place de la mesure en génie logiciel parce qu'il a été construit à partir de huit aspects de la mesure, c'est-à-dire, un aspect général ainsi que les sept raisons de mesurer identifiées lors de la revue de littérature : pour la compréhension, pour l'évaluation, pour l'expérimentation, pour le contrôle de projet, pour l'amélioration du processus, pour l'amélioration du produit et, finalement, pour les prédictions. Ce sont des thèmes très pertinents dans le domaine de la mesure en génie logiciel et qu'il sera très intéressant d'approfondir. Avec ce questionnaire et cette liste d'incertitudes que nous avons produits, nous pouvons dire que l'objectif principal et le sous-objectif ont tous les deux été atteints.

6.3.1 Apport à des recherches subséquentes

Nous avons identifié une série de sujets auxquels notre recherche pourra apporter quelque chose soit comme point de départ, soit comme intrant. Ce sont les sujets suivants :

Concernant la place de la mesure en génie logiciel :

- Recherche d'autres incertitudes dans d'autres sources : autres livres sur la mesure, études portant spécifiquement sur la mesure afin de continuer l'identification des incertitudes de la mesure.
- Exploitation des incertitudes identifiées afin d'expérimenter et de vérifier par des études de cas l'importance du problème sous-jacent à l'incertitude identifiée. Afin, également de penser aux possibilités d'amélioration de l'utilisation de la mesure en génie logiciel.
- Interrogation d'un plus grand nombre de personnes sur la question de la place de la mesure en génie logiciel
- Rédaction d'articles sur la place de la mesure en génie logiciel

Concernant les principes fondamentaux :

- Analyser de manière plus approfondie les résultats quantitatifs des trois études
- Analyser les résultats à partir d'autres points de vue que celui de la mesure
- Utiliser d'autres techniques de vérification des principes fondamentaux
- Participer à d'autres efforts d'identification des principes fondamentaux du génie logiciel (ex. la liste d'envois SEPrinciples)
- Produire des textes explicatifs pour la liste actuelle de principes candidats afin de faciliter la compréhension de ces principes.
- Rédiger des articles sur le thème des principes fondamentaux

6.3 FORCES ET LIMITES DU TRAVAIL

6.3.1 Forces du travail

TROIS ÉTUDES SUR LES PRINCIPES FONDAMENTAUX

L'une des forces de cette recherche est que trois études différentes ont été réalisées afin d'identifier les principes fondamentaux du génie logiciel. La première étude Delphi a permis d'obtenir une liste de principes. La deuxième étude Delphi et le sondage ont permis d'affiner cette liste de principes.

LA QUALITÉ DE L'ÉCHANTILLON DE PARTICIPANTS AUX ÉTUDES SUR LES PRINCIPES FONDAMENTAUX

Nous pensons que l'échantillon de participants aux trois études sur les principes fondamentaux est représentative de la communauté du génie logiciel. Les participants proviennent de nombreux pays. Ils ont des *background* très variés, que ce soit au niveau des études, du domaine d'études ou du domaine d'affaires de leurs employeurs. Le nombre d'années de pratique de certains participants est très imposant, nous portant à croire qu'ils ont beaucoup d'expérience en génie logiciel. Tous ces atouts nous font croire que notre échantillon est de qualité.

6.3.2 Limites du travail

VÉRIFICATION FORMELLE DES RÉSULTATS

Notre liste d'incertitudes a été tirée des commentaires des participants, mais il n'y a pas eu d'expérimentation et de vérification formelle des résultats. Nous avons identifié ces activités comme travaux subséquents à notre recherche.

APPORT DES PARTICIPANTS

L'apport des participants était sommaire et représentait leur opinion. Une interrogation plus approfondie de participants est recommandée pour des travaux futurs. De plus, il serait nécessaire de vérifier que l'opinion des participants représente la réalité, ce qui rejoint le point précédent, à savoir qu'il est nécessaire d'avoir une vérification formelle des résultats.

UTILISATION DE COMMENTAIRES SUR LES PRINCIPES FONDAMENTAUX POUR IDENTIFIER LES INCERTITUDES DE LA MESURE.

Les deux études Delphi et le sondage ont été réalisés principalement dans le but d'identifier les principes fondamentaux. Dans le chapitre sur la problématique, nous expliquons le pourquoi de cette recherche de principes fondamentaux. Cependant, vu la richesse des commentaires de la première étude Delphi, nous avons pensé les utiliser pour des activités autres que celles concernant uniquement les principes fondamentaux. L'identification de la place de la mesure en génie logiciel constituait l'une de ces activités. Cela signifie que les commentaires utilisés pour notre recherche n'ont pas été émis par les participants dans le but d'identifier la place de la mesure en génie logiciel. Selon nous, cela constitue une limite à notre recherche. Cependant, nous pensons que ces commentaires constituaient une bonne source de départ pour l'identification des incertitudes de la mesure en génie logiciel. De plus, nous comptons étendre notre recherche dans le domaine à d'autres sources.

ABSENCE DE TEXTES EXPLICATIFS POUR ACCOMPAGNER LES PRINCIPES FONDAMENTAUX

Au cours des études Delphi et du sondage, nous avons observé que la présence d'un court texte explicatif aurait facilité la compréhension des principes fondamentaux candidats. D'ailleurs, certains participants nous ont fait la remarque. Cependant, la rédaction de tels textes ne pouvait être faite par nous car elle nécessite une forme de consensus auprès des experts en génie logiciel. Nous prévoyons mettre sur pied un atelier, lors d'une conférence internationale sur le génie logiciel, afin de procéder à la rédaction de ces textes.

6.5 APPRENTISSAGE

Mes objectifs personnels étaient d'acquérir une meilleure connaissance du génie logiciel et de l'utilisation qu'on y fait de la mesure. J'ai atteint ces objectifs. En particulier, les commentaires sur les principes fondamentaux m'ont donné une vision plus réaliste du génie logiciel. Ils m'ont aussi permis de mieux comprendre les problèmes auxquels sont confrontés les personnes en génie logiciel ainsi que de prendre conscience d'éventuelles solutions pour arriver à résoudre ces problèmes. D'autre part, l'aspect mesure de mon travail m'a permis d'acquérir de bonnes connaissances sur l'utilisation de la mesure en génie logiciel. Ces connaissances acquises au cours de ma recherche me serviront certainement dans mon cheminement de carrière qui se veut en génie logiciel.

Un autre objectif personnel était de contribuer, tant soit peu, à la définition du génie logiciel en tant que branche du génie. Je crois que cet objectif a également été atteint et que notre recherche donnera suite à d'autres démarches sur le même thème.

CONCLUSION

La motivation de notre recherche était de mieux définir la discipline du génie logiciel. Nous avons vu que cette discipline est encore jeune et, de ce fait, qu'elle est confrontée à plusieurs problèmes. D'abord et avant tout, nous nous sommes demandé si le génie logiciel constituait réellement une branche du génie. Nous avons également mentionné les problèmes d'incohérence dans les normes et ceux de savoir réellement ce qu'est le génie logiciel et quelles sont les qualifications requises pour être ingénieur en logiciel. D'autres problèmes sont directement liés à la mesure, par exemple, déterminer sa place au sein des principes fondamentaux du génie logiciel, son rôle au sein du génie logiciel comparativement à celui qu'elle occupe dans les autres disciplines du génie, s'entendre sur l'importance de la mesure en génie logiciel. Finalement, il y a les problèmes de mise en œuvre de programmes de mesure et de mésententes concernant ce qui doit être mesuré, les outils de mesure et la validité de ces outils.

Au cours de notre recherche, nous avons fait une analyse de la place de la mesure au sein du génie logiciel. Nous pensons qu'une telle analyse a permis d'identifier plus clairement certaines lacunes du génie logiciel. Cependant, notre travail s'est limité à produire un questionnaire permettant d'y analyser la place de la mesure. Pour ce faire, nous avons utilisé les résultats de recherches sur les principes fondamentaux du génie logiciel.

Nous avons établi le questionnaire sur la base des différentes raisons de mesurer identifiées dans notre cadre conceptuel (voir la section intitulée POURQUOI MESURER EN GÉNIE LOGICIEL à la page 13 de ce document) ainsi que sur les commentaires effectués par les participants aux deux études Delphi et au sondage.

Pour arriver à produire ce questionnaire nous avons commencé par établir un cadre conceptuel dans lequel nous avons analysé l'utilisation de la mesure dans le génie tout au long de son histoire et son rôle au sein de différentes disciplines du génie. Puis, nous avons effectué une revue de littérature qui nous a permis d'identifier sept raisons de mesurer en génie logiciel.

Notre cueillette de données s'est faite par le biais de deux études Delphi et d'un sondage qui nous ont donné accès aux opinions d'un grand nombre d'experts en génie logiciel. Lors de la première étude Delphi, nous avons recueilli l'opinion de quatorze experts en génie logiciel. Trente et un

responsables du TCSE ou des comités éditoriaux de l'IEEE Software et de l'IEEE Transactions on Software Engineering ont participé à notre deuxième étude Delphi Quand au sondage, il a été envoyé à près de 4000 membres du Technical Counsel on Software Engineering (TCSE) de l'IEEE, dont 574 ont répondu. Nous avons effectué l'analyse de données à la fois quantitatives et qualitatives. Ces données nous ont permis d'avoir une idée de la place de la mesure en génie logiciel.

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

Notre recherche des incertitudes sur la mesure nous a donné un aperçu de la réponse à certaines des questions que nous nous sommes posées. Par exemple, nous avons pu remarquer que les membres de la communauté du génie logiciel ne semblaient pas s'entendre sur l'importance à accorder à la mesure. De plus, la mesure ne semble pas y être utilisée aussi fréquemment que dans les autres disciplines du génie. Il y a probablement un manque de maturité dans l'utilisation de la mesure en génie logiciel. Par ailleurs, dans les cas où la mesure est effectivement utilisée en génie logiciel, le processus semble manquer de clarté et de rigueur. Il y a donc beaucoup à faire pour améliorer la situation.

Ce qui nous a particulièrement surpris dans les commentaires, c'est le manque d'accord des participants concernant l'importance à accorder à la mesure en génie logiciel. En effet, les participants à nos trois études viennent d'une communauté appuyant fortement le génie logiciel. On aurait alors penser qu'ils seraient tous pour l'utilisation de la mesure. Ce n'était pourtant pas le cas. En fait, on pourrait même parler d'une véritable controverse concernant l'utilisation de la mesure en génie logiciel. Controverse qui s'est d'ailleurs répétée dans les trois études sur les principes fondamentaux. D'un côté, il y a ceux qui pensent que la mesure doit faire partie intégrante du génie logiciel et, de ce fait, que sa place est importante. De l'autre côté, il y a ceux qui pensent que la mesure aurait effectivement un rôle à jouer en génie logiciel mais que, dans son état actuel, il vaut mieux ne pas trop l'utiliser.

En plus de nous avoir permis de comprendre la place qu'occupe la mesure en génie logiciel, notre travail pourrait donner naissance à plusieurs autres projets de recherche sur les thèmes de la mesure et des principes fondamentaux du génie logiciel. En effet, les résultats des trois études sur les principes fondamentaux pourraient être approfondis. Le questionnaire sur la mesure pourrait

permettre d'explorer chacune des incertitude. Plusieurs autres recherches ont été identifiés dans la section sur l'apport de cette démarche à des recherches subséquentes.

Nous espérons que cette recherche sur la place de la mesure au sein des principes fondamentaux du génie logiciel aura contribué à mieux comprendre les problèmes de mesure en génie logiciel. De plus, si les recherches subséquentes identifiées sont effectuées, nous espérons qu'elles permettront d'améliorer le niveau de maturité 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

Définition			
Motivation	**Objet**	**Objectif**	**Utilisateurs de la recherche**
Mieux définir la discipline du génie logiciel	La mesure en génie logiciel	• Produire un questionnaire sur la place de la mesure en génie logiciel • Identifier les incertitudes concernant la mesure en génie logiciel	• Praticiens en génie logiciel • Chercheurs sur la mesure en génie logiciel

Planification		
Étapes du projet	**Intrants**	**Livrables**
Deuxième étude Delphi	• Première étude Delphi • experts des comités IEEE en génie logiciel	Analyse des données recueillies (scores et commentaires) des participants
Rédaction de la proposition	Revue de littérature sur le génie, le génie logiciel, la mesure en génie logiciel, les principes fondamentaux du génie logiciel	Proposition
Sondage	• praticiens du génie logiciel, membres des comités IEEE • Delphi 1, Delphi 2	Analyse des données recueillies (recherche de *patterns*, application de tests statistiques)
Rédaction du rapport final	• 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	Rapport final

<table>
<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>Scores et commentaires de 31 participants sur les principes fondamentaux</td><td>Recherche d'information sur les sujets suivants:
• Génie,
• Génie logiciel,
• principes fondamentaux du génie logiciel,
• mesure en génie logiciel</td><td>Scores et commentaires de 574 participants sur les principes fondamentaux</td><td>• Calcul de moyenne, médiane, écart-type et consensus
• Analyse de la corrélation entre les différents principes
• Présentation des résultats des trois études</td><td>• Analyse d'incertitudes de la mesure en génie logiciel
• Lien avec ce qui a été retenu de la revue de la littérature
• Liste d'incertitudes de la mesure en génie logiciel</td><td>Liste de questions sur les incertitudes de la mesure en génie logiciel</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>Travaux subséquents</th></tr>
<tr><td colspan="2">• Contexte statistique :
- Delphi 1 : 14 participants
- Delphi 2 : 31 participants
- Sondage : 574 participants
- Principe A : Coté parmi les plus faibles. Peu de consensus sur le score alloué à ce principe
• Les objectifs ont été atteints
• Les réponses des participants ont permis d'identifier de nombreuses incertitudes quant à l'utilisation de la mesure en génie logiciel
• Cette recherche pourrait contribuer à mieux définir la discipline du génie</td><td colspan="2">Positif :
• Trois études différentes ont été réalisées
• L'échantillon est de qualité
Négatif :
• Il n'y a pas eu d'expérimentation et de vérification formelle des résultats
• L'apport des participants est sommaire et représente leur opinion</td><td>Concernant la place de la mesure en génie logiciel :
• Recherche d'incertitudes dans d'autres sources
• Exploitation des incertitudes
• Interrogation d'un plus grand nombre de personnes
• Rédaction d'articles sur le sujet
Concernant les principes fondamentaux
• Analyse plus approfondie des résultats des trois études
• Analyse des principes fondamentaux selon d'autres points de vue que celui de la mesure
• Participer à d'autres efforts d'identification des principes fondamentaux du génie logiciel
• Rédaction de textes explicatifs devant accompagner les principes fondamentaux
• Rédaction d'articles</td></tr>
</table>

APPENDICE E

COMMENTAIRES LIÉS À LA MESURE

Dans cette annexe, nous avons regroupé tous les commentaires qui étaient liés à la mesure.

Commentaire 1 :
Measure what matters, not what's handy. Predict what to expect from the measurements before taking them, and use the results to guide adjustments.
Discussion : Measurement is important, but it should deliver value commensurate with the cost of collecting and analyzing the information. The choice of what aspects of a project (both process and product) to measure should be driven by the project's objectives. The conventional measurements may or may not be relevant; be selective, as measurement requires resources.
Sometimes carefully stated qualitative expectations can serve you better than counting something peripheral.
Measurement should be preceded by prediction or estimation, so that the results can confirm or refute established expectations.

Commentaire 2 :
Goal of software engineering (management) is Quality improvement, Cost Down (Productivity improvement) and Deliver in time. But continuous efforts for improving environment, motivating and educating people (engineering staff), and improving process is more important.

Commentaire 3 :
Quality Requirement must be specified for all quality characteristics, and quality must be measured for all requirements.

Commentaire 4 :
Software Quality Evaluation must be Scientific and quantitative. But it must be practical and cost effective.

Commentaire 5 :
Adopt a managed improvement-oriented 'production' cycle :
Like any complex endeavor, software development must be managed in a more controlled style than the ad hoc (almost chaotic) approaches adopted in the past. This should incorporate something like the following steps : plan, act, monitor, control, measure, improve. An integral component is the use of product and process metrics.

Commentaire 6 :
Quality should be built in to product and process :
Again, fairly obvious. Attempting to 'add on' quality after software components have been built is (a) unlikely to be fully successful and (b) far more expensive. Similarly, trying to shore up a badly managed and executed software process after the event is too late.

Commentaire 7 :
Every software project should set a reliability objective or objectives for its deliverable product(s). Actual reliability achieved should be estimated at various points in the project and compared with the objective(s), the ratio being used to manage the project.

Commentaire 8 :
Every software project should develop an operational profile or profiles in consultation with users or their representatives, so that expected usage can be quantified and development and testing focus placed on the most used and/or most critical functions.

Commentaire 9 :
Every project should plan and quantitatively specify its reliability strategies (the blend of fault prevention, fault removal, and fault tolerance) based on its reliability objective(s).

Commentaire 10 :
Projects should choose their tools and methodologies based on quantitative criteria, measuring their effect on reliability, schedule, and cost with respect to cost of implementation.

Commentaire 11 :
CONTROLLING RISK. There are large quantities of uncertainty, and risk of deviation from plans, in any project. You cannot eliminate risk, but you can document it, plan and design for it, accept it, measure it, and reduce it to acceptable levels.

Commentaire 12 :
QUANTIFICATION MANDATORY FOR CONTROL. The multiple concurrent quality and cost demands of most systems, means that a quantified and testable set of requirements is necessary, to get control over quality and costs.

Commentaire 13 :
Accept uncertainty in estimates. No estimate is correct unless it be by a lucky guess. Assume that an estimate will be incorrect, more underestimated than overestimated, and put aside contingencies.

Commentaire 14 :
Quantitative when possible, otherwise qualitative. Better the right qualitative assessment than the wrong quantitative one.

Commentaire 15 :
Would add "for project management and technical development"

Commentaire 16 :
But beware faked or make-up numbers.

Commentaire 17 :
This is one of the only mechanisms we have by which we might hope to improve our processes to something resembling an engineering discipline.

Commentaire 18 :
Desirable but not always practical.

Commentaire 19 :
Is control important? If so, how important is it? Is it more important than getting the right answer. Besides, getting 'control' is a goal in my opinion not a principle.

Commentaire 20 :
"as formally as possible" ... but no more formally. Again, qualitative results are also respectable and may be more relevant.

Commentaire 21 :
As formally and quantitatively as.

Commentaire 22 :
Don't strongly disagree!

Commentaire 23 :
Need to use quantitative measures wherever it is possible and practical.

Commentaire 24 :
In relation to one of the comments, to me control is certainly more important than getting the right answer - how do you know if it's right and how can you use it if you have no control?

Commentaire 25 :
Not necessarily always quantitative!!!

Commentaire 26 :
But set them measurably and testably is far more crucial.

Commentaire 27 :
I consider that decision making, even in the SE context, will often require consideration of qualitative as well as quantitative data.

Commentaire 28 :
There is a place for creativity, chance, and risk within engineering.

Commentaire 29 :
There are too many issues raised for a single rating to be appropriate. I do not believe SE predictions and quantitative measures are mature enough or reliable for this to be any more than a pious hope. We should expect SE managers to use decision-making criteria at least as good as management of other engineering activities (which are not significantly better than SE for novel or large projects).

Commentaire 30 :
For many important decisions this principle assumes a static or predictable situation. Political criteria and educated guesses about the future may have more importance. When this principle should be used, reliable quantitative data are hard to come by.

Commentaire 31 :
I don't understand the question. "Decision making" is a very broad term. I think quantitative measures should apply to some kinds of decisions, but as worded, the question is too broad to response to.

Commentaire 32 :
I would change the emphasis in the explanation somewhat, as follows : Decision-making in software development must be based on quantitative data. This data should not only guide the management of the projects themselves, but should enable senior management to gauge the progress of process improvement. It should provide the basis for the selection of process methodology and the tools to implement a methodology.

Commentaire 33 :
I strongly agree with this principle. If software becomes an engineering discipline, then it will require objective metrics to base decisions and measure performance.

Commentaire 34 :
While I am a proponent of quantitative measurement, decisions should be based on the amount of information that is cost-effective to acquire given the importance ($) of the decision, and non-quantitative data should not always be excluded as is implied by the first sentence.

Commentaire 35 :
The need to measure is a sign that you don't understand the problem/artifact/technology. Measurements can be useful in such situations, but figuring out which axes of comparison to weigh more heavily must also be recognized as heuristic, not scientific, in nature.

Commentaire 36 :
Agree with principle, "apply and use measurement in decision making." (note that I left the word, quantitative out). I don't agree with the rest. For example, a choice of tools maybe based more on market analysis than a technological (scientific) analysis. Also software management and implementation decisions should incorporate business, organization, product and technology goals. I think the larger principle is "As much as possible, make software decisions based on a full analysis of the facts, allowing for uncertainty." Facts can range from opinions massaged by Delphi techniques to statistical analysis of data. If that was the principle, I would of course give it a 10.

Commentaire 37 :
On many projects an attitude of "I liked this tool, I know how to use it, let's go!" is perfectly adequate. Fundamental principle? I don't think so.

Commentaire 38 :
While qualitative information is also important, most IT executives want some level of quantitative information to base their decisions. As for application to choice of tools, methods, and PM, the situation would have to dictate the approach.

Commentaire 41 :
Statistical Quality Control measures for the software product and process (i.e., ISO 9000-3 and ISO 9001).

Commentaire 42 :
This is a bit like prunes (2 is not enough and 8 is too many!) in that there are situations where the cost/benefit for more assessment does not buy more quality. In general, I agree with Demming's principles and apply them. I also strive to eliminate non-contributing activities from the process where I can. It's a balance principle.

Commentaire 43 :
This would be fine, if software quality could be defined and measured.

Commentaire 44 :
I agree if quality objectives are based on adherence to an organization's process (as measured by objective metrics, quality control, etc.) But it takes a very mature software organization to set quality objectives.

Commentaire 46 :
I agree with most of this question. I don't think it's always possible to "quantify" expected usage. Simply identifying usage patterns is an important step. I'm not sure that "quantifying" them would produce much additional benefit in most cases. Again, this question muddles an objective (simplifying goals sets) with an implementation (quantifying expected usage).

Commentaire 47 :
This is another case of the prune analogy. If the material is readily available, and the time to study is reasonable to get good results, do it. If, on the other hand, it becomes a constraint to progress, the ROI must be examined.

Commentaire 48 :
Uncertainty certainly exists in software engineering, as in the world at large. But much of the uncertainty in software development is avoidable if the industry were in a position to pay more attention to front-end tasks, such as establishing feasibility, devising a core architecture, identifying and reducing risks, producing a construction plan, and estimating costs and time schedule, before rather thoughtlessly embarking on construction. Therefore, treating uncertainty as an inescapable fact of software life is a dubious principle to proclaim. Yet, there is the hint of a principle here. It needs to be expressed more in terms of identifying and reducing uncertainty than in terms of accepting it.

Commentaire 49 :
While my rating is only a point away from the median, I think it is important enough to recognize that things get done that are measured. This is especially true for decisions. Measuring the progress and the performance of software processes is essential for deriving improvements needed for meeting industry demands.

Commentaire 50 :
I'm shocked that the median on this is so low. This is a *fundamental* principle of all other engineering disciplines, why not *software* engineering!! If it is not a fundamental discipline, take the word *engineering* out of software engineering. It's not a question of *how* mature quantitative methods are, they need to be applied. It's through the application of quantitative methods and the experience gained and mistakes made) that we mature these methods. An engineer can (and should) always provide a qualification along with a measurement as to its "believability" when presenting it to decision makers, but must still strive to back up what he/she recommends with quantitative information.

Commentaire 51 :
But certainly not exclusively or for excessive cost.

Commentaire 52 :

Statistical Analysis of management and technical performance. Quantification is necessary to fully describe the life cycle use of the system or product(s) within the environment(s) the customer anticipates for their needs. Failure to do so denies the potential software engineer from being recognized by other engineering disciplines.

Commentaire 54 :
Too many caveats for a good fundamental principle.

Commentaire 55 :
No There is a place for creativity, chance, and risk within engineering.

Commentaire 56 :
I feel this is essential.

Commentaire 57 :
Sometimes gut feeling leads to the best business decisions, because we don't know which characteristics to quantify.

Commentaire 58 :
What else would you base decisions on? belief? Intuition (that is bound to happen anyway).

Commentaire 59 :
We simply don't know enough to be quantitative. If you go out there in the "real world" you have difficulty even getting people to believe that they need to understand the problem!

Commentaire 60 :
Not universal applicable, nor universally desirable.

Commentaire 61 :
System Performance Requirements allocated to software architectural components. Abstraction as a generalization method can help. The danger is in abstraction of the system or product(s) without clear quantification of the environment(s) and behavior(s) of the system or product(s) during customer life cycle use.

Commentaire 62 :
To my mind, this is the same as saying "Travel to San Francisco using the optimal route". You'd choose something that wasn't? The real issue is the criteria on which this "flexibility" requirement is measured. I remain of the opinion that this is an open door for hackers.

Commentaire 63 :
Quantifiable management and technical performance requirements allocated to the software product(s).
Quantification of the problem by other engineering disciplines should be considered, as well as the associated previous solutions identified by other engineering disciplines.

Commentaire 64 :
Statistical Quality Control measures for the software product and process (i.e., ISO 9000-3 and ISO 9001). This could be consolidated with Fundamental Principle A below.

Commentaire 65 :
Quantification of the process and product quality is necessary during the quantification of the product performance requirements for the life of the system or product(s). This is related to Fundamental Principle A below.

Commentaire 66 :
This must consider the measurements discussed in Fundamental Principle A below. The tradeoffs in any engineering product must consider the relationships between cost, schedule, technical product performance, and the quality of the processes and the end product.

Commentaire 67 :
Uncertainty is created by the lack of engineering and scientific discipline for the definition of the problem, or services to be provided by the system or product(s). To become an engineering discipline, the future software engineer must be taught the same engineering and scientific fundamentals concerning systems and products. This means clear quantification of the performance required by the process and results provided by the associated system or product(s).

Commentaire 68 :
I believe that uncertainty is greater in software engineering than in other engineering disciplines because software production relies on human expertise. For example estimates of production effort and schedule are very inaccurate i.e. include a large amount of uncertainty. The IEEE Code of ethics states that we agree 'to be honest and realistic in stating claims or estimates based on available data'. It would be dishonest to pretend we don’t have major problems producing software due to our uncertainty about the behavior of products and processes as a result of the impact of humans on the development process.

Commentaire 69 :
However, this principle sounds like motherhood. If software was predictable on get-go, then we would not need engineering, would we? This principle should be combined with principle A to give it meat. Managing uncertainty involves collecting facts, postulating future events, assigning probabilities and impact and designing contingencies: these activities are part of a bigger principle involving decision making.

Commentaire 70 :
Instead of narrowly focusing on uncertainties as this principle does, software engineers must recognize that each decision they make is based on assumptions about future contingent events (with differing degrees of uncertainty) and they must weigh these uncertainties based on available facts - hopefully quantified ones. The above principle does not integrate uncertainty to disciplined decision making - the latent principle behind principles O and A. With disciplined decision making (PSP is an example) we can link the social/economic aspects of software engineering to other disciplines.

Commentaire 71 :
Identifying rigorous quantitative and procedural techniques for software engineering is necessary to improve the state of the art. The practice of software engineering, however, will always remain a combination of a art and science. The codifying of practices will never replace the intuitive leaps that are indicative of the masters of the craft.

Commentaire 72 :

Principle A needs to be complemented with "use experience and guts feeling as part of your decision process"

Commentaire 73 :
Need definition of terms. For instance 'quality' (still) means different things to different people.

Commentaire 74 :
1)Measure the individual and team knowledge capacity in software production. (9) 2)Measure the potential impact of software artifact in economy and industry.

Commentaire 75 :
I read the first principle and got the impression that quantitative measurements would be the decision making tool. After thinking on it, perhaps the principle should read: Identify and use quantitative measurements to assist in decision making. Many times in SW development, when measurements can give misleading indications. Decision making is a human process; the measurements are tool to be used in the decision making process.

Commentaire 76 :
don't forget the subjective areas of project management, personnel, rewarding initiative; diversity in the workplace (minorities, women). Not every SWE issue can be quantified, and numbers can lie anyway... IE, I'm a big product metrics fan, but if used improperly, the value of the metrics are destroyed. Same with PSP and CMM.

Commentaire 77 :
Ad managing uncertainty: proactive measures are costly; it is cheaper to implement an effective risk management system which is able to provide effective REACTIVE measures.

Commentaire 78 :
If software engineering is supposed to be "quantifiable", it is not clear that _any_ of these statements are "fundamental," because nothing quantifiable can be derived from them. Most are simply platitudes that apply to software development in general; it is impossible to disagree with something like G or O. The ones that seem to address some quantifiable management objective like ``quality'' beg the question of how that objective is to be measured, or what to do in the case of the many kinds of software where quality either cannot be measured in any meaningful way, or is of negligible importance compared to other considerations.

Commentaire 79 :
Don't get hung up on measurement. Focus on the skills and continuous training of the PEOPLE who represent your critical capital assets.

Commentaire 80 :
All the statements listed are good practices, but implementation depends on the situation. Defining a process and measuring results seem to be most important for "engineering" and are vital for proper management.

Commentaire 81 :
A needs to be clarified. If A. does not imply that decision-making relies solely on quantitative measurements, then I shall give a score of 10.

Commentaire 82 :
A and B are lofty goals which we do not understand well so I don't want to subscribe to them as they are currently practiced.

Commentaire 83 :
I've had quite a few problems trying to quantify the Principles given above - mainly because the wording implies some overlap, and the purpose of each principle is not defined and is not obvious. First, I must point out the software engineer has to deal with two phyla of knowledge - a) that of devising and managing the process which supports the creation of this component, and b) that of engineering the software component. The Principles seem to confuse the two issues, but they are best considered separately. a) In the creation and management of the process, the aim is to help the project to correctly complete the task in the swiftest and most accurate manner. Metrics are very important here but they should not be onerous to gather and must not slow down or distract the engineering of the component. It is important to note that these metrics are of most use in retrospective analysis of a project in order to improve the process or allocation of resources for the next project - they rarely help the project already under way. Unfortunately, this means that usually metrics are kept in an ad-hoc manner and cast into a void at the end of the project. Projects seldom learn from another's mistakes and this must change in any well-engineered process. Some of your Principles (A,E,H,K) appear to be aimed at improving the engineering process - collecting enough information about the process to enable analysis and improvement. b) In the creation of the component, software engineering proceeds in four main distinguishable phases: o Specification - in which, for simplicity, I include both Requirement or problem specification and the specification of the solution). o Design - where a system which meets the specification is fleshed out o Implementation - the detail of the Design is finished off and implemented. This phase is usually not recognized nowadays since in software, with adequate compilers, a Design can generate what used to be called the Implementation. Nevertheless, the distinction is important - during the Design phase the system may not be stable but during Implementation the Design does not change. o Test - where the complete system is tested in the environment of which it is a component. Note that this does not mean that no "testing" is done until the end. See below. Concurrently, during these phases, Validation and Verification must be carried out, partly by the engineers involved in the phase processes and partly by other engineers. The system under construction must me checked to make sure that it meets its requirements ("validation"?) and that it works under all conditions ("verification"?). Leaving this until the late implementation phase can be disastrous ! Again, some of the proposed Principles seem aimed at making sure that a phased process is used (C,J), that each phase is optimal (G,I,M,N), and that the software can be checked at all levels of the process (D). Comments on some of the principles follow : A : This can only be done from measurements of previous projects, and then mostly in choosing the best methodologies and best use of resources - e.g. a little extra time in specification when using object orientation pays dividends during later phases. B : I CANNOT UNDERSTAND ANYBODY WHO SAYS REUSE IS NOT IMPORTANT. In my opinion, a good software design and implementation automatically has reusable components. Design for Reuse does not mean adding extra dongles not needed by the current project, it means rational and useful componentisation ! It means appropriate and beautiful encapsulation. A good measure of any design is how much it reuses its own components, and how much it is reused - what used to be called "Structured Design". Additionally, reuse can happen at all levels - a specification can be reused just as well as an implementation. C & I : These seem to me to be connected. The management of complexity is always difficult and the effective use of hierarchy and encapsulation (componentisation) is vital.

Commentaire 84 :

Don't apply quantitative measures in decision making when proofs are better. For example, its easily proven that strictly relational data management will produce one of : inflexibility, poor performance, unnecessary administration or limited functionality in certain cases where networked data solutions are required to complement relational solutions. Don't rely on measurements to reach such a decision to use data solutions that are outside of the domain of the relational database model.

Commentaire 85 :
The practice of software development needs far less theory and much more observation, measurement, and evaluation of what is and is not successful in use. This will be costly, difficult and time consuming, which is probably why it has not been done so far. If we want to consider ourselves part of the engineering profession, we must root ourselves firmly in the practical world and also develop a strong sense of self-discipline.

APPENDICE F

DONNÉES DÉMOGRAPHIQUES SUR L'ÉCHANTILLON DU SONDAGE

Nous vous présentons quelques données démographiques sur l'échantillon des participants au sondage sur les principes fondamentaux.

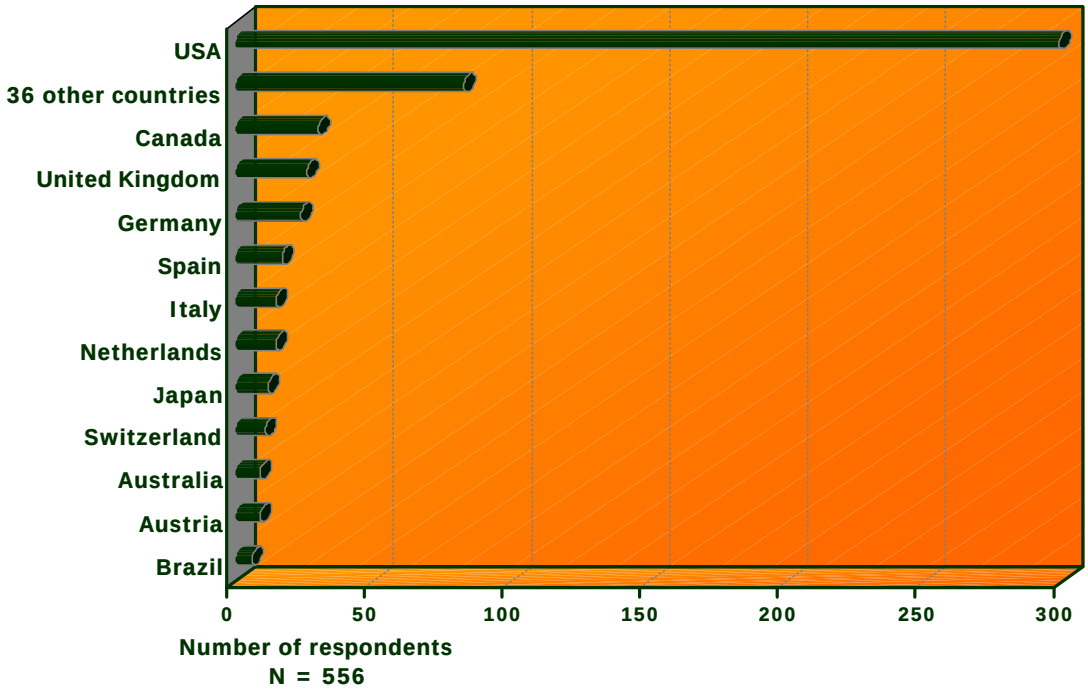

Figure F.1 Répartition par pays

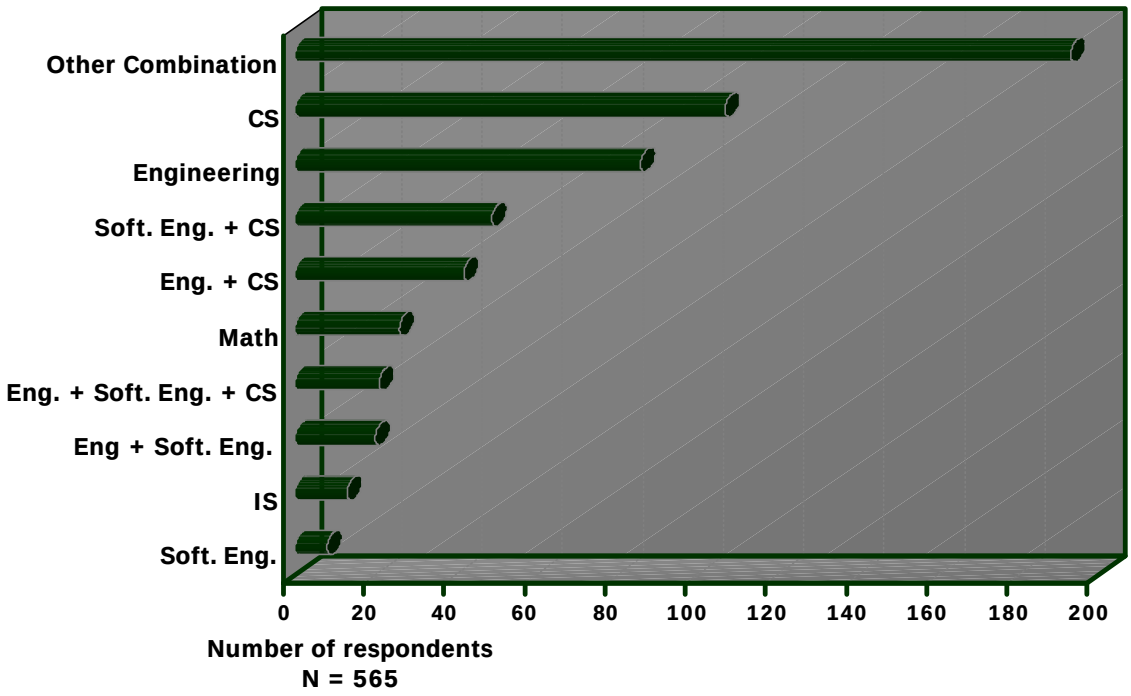

Figure F.2 Répartition par domaines d'étude

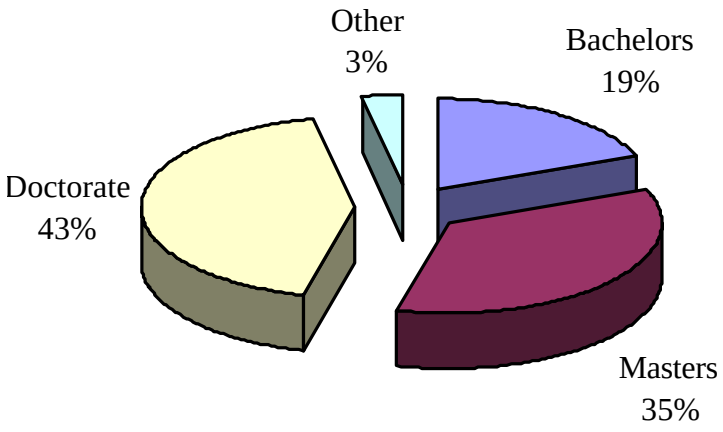

Figure F.3 Répartition par niveau d'étude (N = 559)

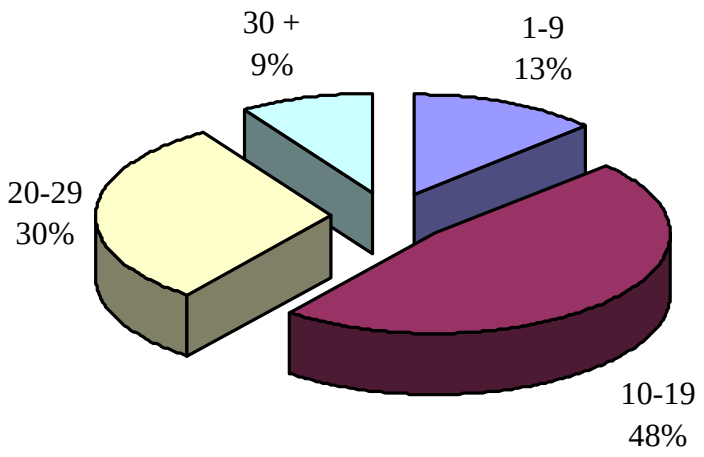

Figure F.4 Répartition par nombre d'années d'expérience en tant que praticien dans l'industrie (N = 558)

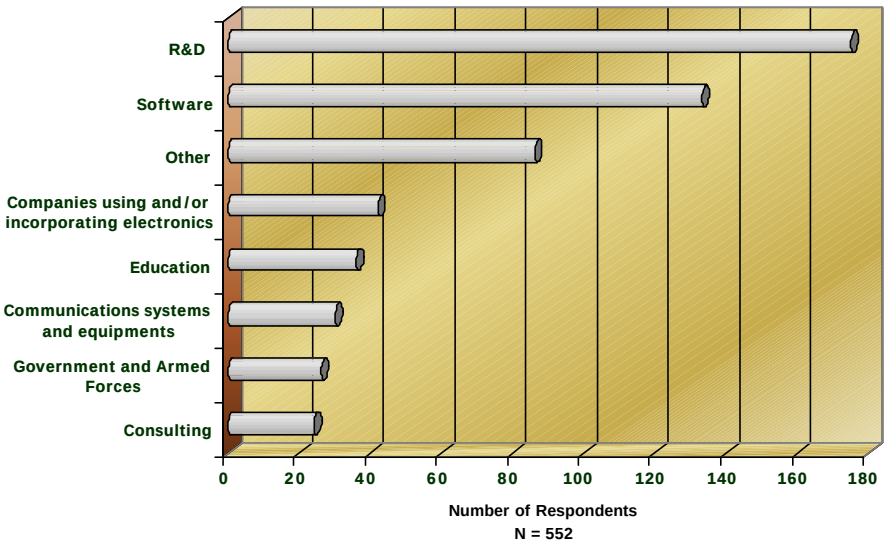

Figure F.5 Répartition selon le domaine d’affaire de l'employeur

RÉFÉRENCES

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

Basili, Victor R., David H. Hutchens et Richard W. Selby. 1986. «Experimentation in Software Engineering», *IEEE Transactions on Software Engineering*, Vol. SE-12, no 7 (juillet), p. 733-743. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.

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

Boehm, Barry W. 1991. «Software Risk Management : Principles and Practices», *IEEE Software*, Vol. 8, no 1 (janvier), p. 32-41. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.

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.

Daskalantonakis, Michael K. 1994. «Achieving Higher SEI Levels», *IEEE Software*, Vol. 11, no 4, (juillet), p. 17-24. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.

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., Robert L. Glass et Shari Lawrence Pfleeger. 1994. «Science and Substance : A Challenge to Software Engineers», *IEEE Software*, Vol. 11, no 4 (juillet), p. 86-95. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.

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.

Grady, Robert B. 1994. «Successfully Applying Software Metrics», *Computer*, Vol. 27, no 9 (septembre), p. 18-25. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.

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*, Vol. 14, no 2 (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.

Software Engineering Institute (SEI). 1995. Carnegie Mellon University, *The Capability Maturity Model : Guidelines for Improving the Software Process*, Massachusetts, Addison Wesley.

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

Stark, George, Robert C. Durst et C. W. Vowell. 1994. «Using Metrics in Management Decision Making», *Computer*, Vol. 27, no 9 (septembre), p. 42-48. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.

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.

Weller, Edward F. 1994. «Using Metrics to Manage Software Projects», *Computer*, Vol. 27, no 9 (septembre), p. 27-33. In *Applying Software Metrics*, sous la dir. de Paul Oman et Shari Lawrence Pfleeger. Los Alamitos (California) : IEEE Computer Society Press.