Principes fondamentaux du génie logiciel : Une étude Delphi

ROBERT DUPUIS[1], PIERRE BOURQUE[2], ALAIN ABRAN[2], JAMES W. MOORE[3]

Le génie logiciel ne dispose toujours pas d'un ensemble de principes fondamentaux reconnus par tous. Cet article décrit une démarche qui vise le développement d'un tel ensemble. Un premier atelier de travail tenu à Montréal en 1996 a permis la formulation de critères que devraient rencontrer de tels principes. Suite à cet atelier, une étude Delphi a été menée avec la participation de quatorze experts internationaux. Seize principes ont donc été proposés et évalués par les répondants. On constate que la gestion du changement a été une grande préoccupation de ce groupe, et qu'elle l'a emporté sur les considérations de discipline, de mesure et de contrôle. Un deuxième atelier tenu à Walnut Creek, CA en 1997, a produit une liste de critères améliorés, ainsi que des formulations plus précises pour quelques-uns des principes.

Introduction

Malgré la tenue de nombreuses rencontres scientifiques, la publication d'une quantité considérable de périodiques, livres et cours sur le domaine, le génie logiciel ne dispose toujours pas d'un ensemble de principes fondamentaux reconnus par tous (Boehm [1983], McConnell [1996, 1997], Brooks [1996]).

Le Software Engineering Standards Committee de l'IEEE[4], sur la recommandation de Parnas [1995] a entrepris le développement d'un tel ensemble en 1996. Un premier atelier de travail s'est réuni à Montréal en octobre 1996 sur ce sujet. On y a produit un ensemble de critères préliminaires et formulé quelques exemples de principes qui les respectaient. Cette rencontre a aussi conclu à l'utilité de mener une étude de type Delphi pour valider ces critères et pour tenter de constituer une première liste de principes fondamentaux (Jabir [1997]).

Cette étude a été menée au printemps 1997 auprès de 14 membres éminents de la communauté internationale du génie logiciel. Une synthèse de leurs suggestions a produit une liste de 16 principes qu'ils ont ensuite évalués. Les résultats de cette étude sont présentés ici. Nous

---

[1] Robert, Dupuis est professeur et directeur de la maîtrise en informatique de gestion et de la maîtrise en génie logiciel au Département d'informatique, Université du Québec à Montréal, C.P. 8888, succ. Centre-Ville, Montréal (Québec) Canada H3C 3P8, Téléphone : 1 514-987-3000 poste 3479, Télécopie : 1 514-987-8477, courrier électronique : dupuis.robert@uqam.ca

[2] Pierre Bourque et Alain Abran sont respectivement directeur adjoint et directeur du Laboratoire de recherche en gestion des logiciels au Département d'informatique, Université du Québec à Montréal, C.P. 8888, succ. Centre-Ville, Montréal (Québec) Canada H3C 3P8, Téléphone : 1 514-987-3000 postes 0315 et 8900, Télécopie : 1 514-987-8477, courrier électronique : bourque.pierre@uqam.ca et abran.alain@uqam.ca

[3] James W. Moore, Special Assistant for the Application of Software Standards and Reuse, Strategic and Theater Army Systems, The Mitre Corporation, 1820 Dolley Madison Blvd. McClean, Virginia USA 22102-3481, Téléphone : 1 703-883-7396 Télécopie : 1 703-883-5432, Courrier électronique : moorej@acm.org

[4] http://www.tcse.org/tcsesesc.html

présenterons d'abord les objectifs qui ont justifié la mise sur pied du projet, puis nous décrirons brièvement le déroulement de l'étude, pour consacrer la part la plus importante aux suggestions et commentaires des panelistes.

Quels principes fondamentaux?

Les normes en génie logiciel souffrent de quelques faiblesses importantes. Une première est la difficulté qu'il y a à transposer dans la pratique un ensemble fort impressionnant de normes (Moore [1997]) édictées par des praticiens compétents et à l'aide d'un processus éprouvé (Coallier [1995], Read [1994], Tripp [1995]). Plusieurs raisons ont été avancées pour expliquer ce manque de pénétration dans l'industrie et nous ne mentionnerons ici que le manque relatif de confiance que les praticiens ont dans les normes (Robinson [1996], Ramamoorthy [1996]). L'existence de principes fondamentaux qui sous-tendraient les normes et qui seraient appuyés par un certain nombre d'expériences rigoureuses pourrait combler en partie cette lacune (Brooks [1995], Glass [1996]).

La Figure 1 illustre la place que pourraient occuper les principes fondamentaux. On y illustre le rôle d'intermédiaire que devraient jouer ces principes fondamentaux entre les principes généraux de disciplines plus larges, ou sous-jacentes, comme l'informatique et la gestion de projets, et les normes en génie logiciel. Les liens bidirectionnels entre ces éléments montrent la volonté d'alterner l'abstraction et la spécialisation entre les principes et la pratique.

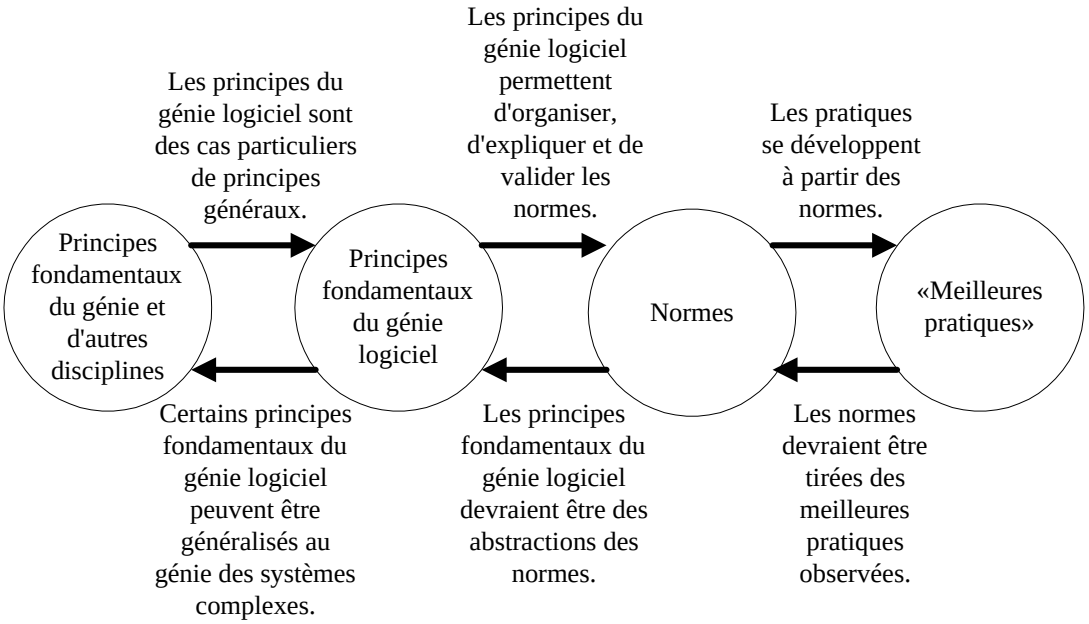

Figure 1 Place des principes fondamentaux

La définition de la discipline en génie logiciel et par conséquent celle des ingénieurs du logiciel demeure toujours une problématique (Shaw [1994], Parnas [1997]). Une réflexion sur les principes fondamentaux du domaine contribuerait à délimiter les activités qui différencient le génie logiciel dans l'ensemble des activités reliées à l'informatique. Dans la même veine, cela permettrait aussi de mieux définir les programmes de formation dans le domaine.

Ces réflexions ont déclenché une démarche de définition de principes fondamentaux au sein du comité des normes en génie logiciel de l'IEEE. Un premier atelier de travail, tenu à Montréal en octobre 1996 dans le cadre de SES'96[5], a produit des critères et quelques suggestions de principes (Jabir [1997]). Les critères spécifiaient que les principes doivent être :

- moins spécifiques que les méthodes ;
- plus durables que les méthodes et les techniques ;
- tirés de la pratique ;
- reliés à au moins un concept sous-jacent ;
- ne pas cacher un compromis ;
- assez précis pour pouvoir être contredits par l'expérimentation.

De plus, cet atelier a recommandé la tenue d'une étude Delphi auprès d'experts internationaux du domaine. La Figure 2 illustre l'ensemble de cette démarche, ainsi que des détails sur l'étude Delphi elle-même.

[5] Les actes du Forum on Software Engineering Standards Issues (SES'96) sont disponibles à http://saturne.info.uqam.ca/Labo_Recherche/Lrgl/ses96.htm

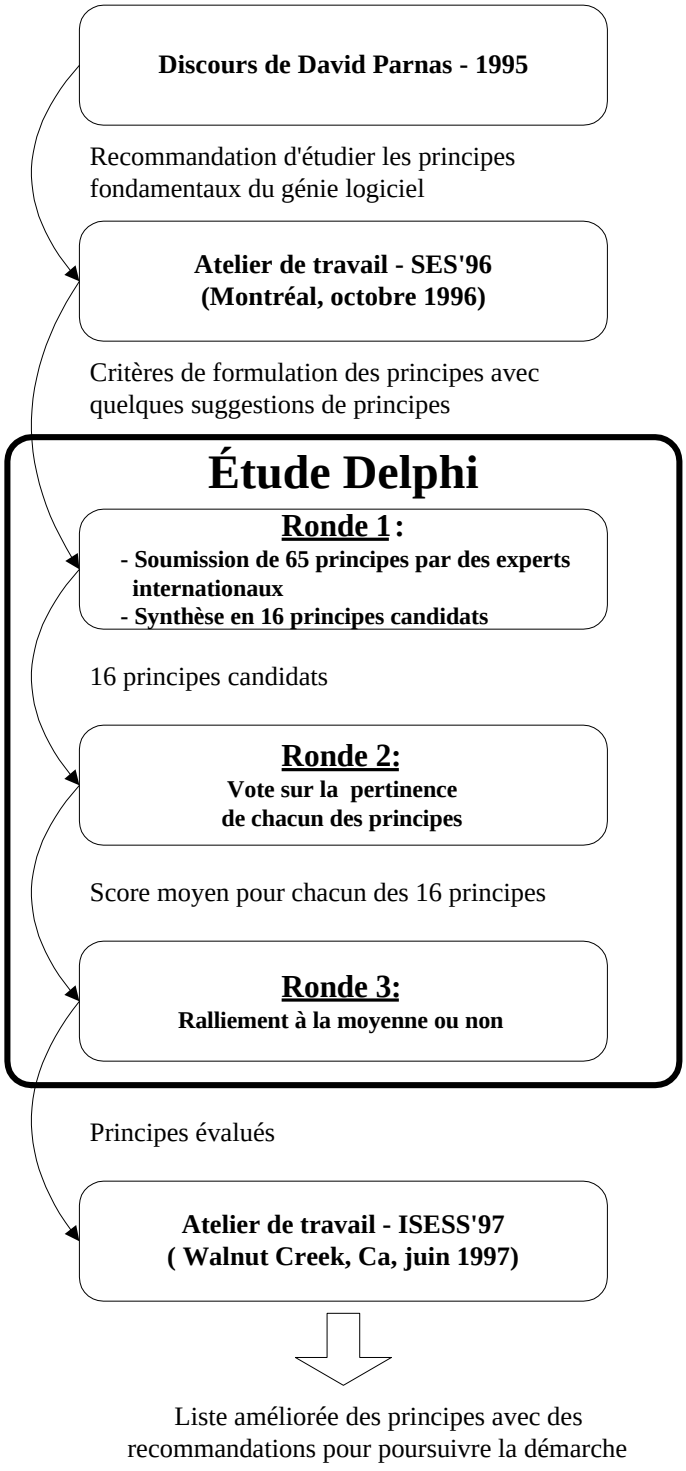

Figure 2 Activités de définition des principes fondamentaux

Déroulement

La volonté de consulter des représentants éminents de la communauté internationale allait de pair avec le choix de la méthode Delphi. Cette technique consiste à former un panel d'experts dont

chacun ignore la composition jusqu'à la fin, afin que le prestige ou le pouvoir d'un autre membre du groupe n'influence pas le cours du débat. On demande dans une première ronde aux individus de soumettre des suggestions. Nous leur avons donc soumis les critères établis à Montréal en octobre 1996 et nous leur avons demandé de produire une liste de cinq principes fondamentaux qui leur semblaient les plus pertinents. Nous avons adressé ce message à 52 personnalités choisies par un sous-groupe ayant participé à l'atelier de Montréal. Treize d'entre elles ont répondu à notre demande et nous avons donc obtenu 65 candidats à des principes fondamentaux.

Deux des coordonnateurs de l'étude Delphi ont chacun de leur côté synthétisé ces 65 suggestions en un certain nombre de principes. La mise en commun de leurs synthèses a produit une liste de 16 candidats à des principes fondamentaux qui respectaient le mieux les critères de l'atelier SES'96, et qui permettaient de prendre en compte 64 des 65 suggestions. Nous avons surtout recherché à cette étape à inclure le plus de suggestions possibles.

Dans la deuxième ronde nous avons demandé aux participants d'évaluer, sur une échelle de 1 à 10, chacun des 16 candidats à des principes fondamentaux produits lors de la première ronde. Nous demandions aussi aux participants de commenter leurs choix, ce que la plupart ont fait.

La dernière ronde d'une étude Delphi vise à consolider et confirmer les consensus qui émergent. Nous avons donc retourné aux participants les scores moyens obtenus par chacun des candidats à des principes fondamentaux, en leur demandant cette fois s'ils se ralliaient à ce score moyen, et d'y aller de nouveaux commentaires s'ils le souhaitaient.

La valeur de ces résultats repose surtout sur l'expertise des répondants. Le Tableau 1 contient la liste publique de 12 des 14 participants[6] :

Répondant	***Organisme***	***Pays***
M. Azuma	Université Waseda	Japon
F.P. Brooks	University of North Carolina	USA
R.N. Charette	ITHABI Corp.	USA
P. DeGrace	Consultant	USA
C. Ghezzi	Politecnico di Milano	Italie
T. Gilb	Result Planning Ltd	Norvège
B. Littlewood	City University	Royaume-Uni
S. MacDonell	University of Otago	Nouvelle-Zélande
T. Matsubara	Matsubara Consulting	Japon
J. Musa	Consultant	USA
R. Pressman	R.S. Pressman & Associates	USA
M. Shaw	Carnegie-Mellon University	USA

Tableau 1 Experts internationaux ayant participés à l'étude Delphi

Deux autres participants ont préféré garder l'anonymat. Nous croyons que cette liste représente non seulement diverses approches au génie logiciel, mais aussi bien des visions théoriques que pratiques, et avec une distribution internationale des experts.

---

[6] On trouvera une courte biographie de chacun à l'adresse http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html.

Les quatre sections suivantes présentent les résultats des trois rondes de l'étude Delphi. Dans les Tableaux 2, 3, 4 et 5, le premier nombre indique le score moyen obtenu selon la pertinence reconnue par les participants, le second indique le nombre de participants (maximum 12)[7] qui admettent que le score moyen représente bien leur opinion. Le maximum possible est donc de 10 pour la pertinence et de 12 pour le consensus.

On observe ainsi trois groupes de principes fondamentaux plus intéressants : un premier de principes fondamentaux très pertinents, un second de principes fondamentaux évalués comme étant beaucoup moins pertinents, et un troisième groupe de résultats surprenants, compte tenu du credo du génie logiciel où la mesure et la discipline occupent une grande place.

Principes évalués comme étant les plus pertinents

Un premier groupe de principes fondamentaux ont obtenu un score moyen supérieur à 8/10 et ont obtenu le ralliement d'au moins 10 des 12 répondants de la dernière ronde. Ces principes sont considérés comme «pertinents» par les participants[8]

PRINCIPE	*PERTINENCE*	*NB. DE PARTICIPANTS D'ACCORD (MAX = 12)*
1. Comme le logiciel est, de par sa nature même, sujet au changement, il faut planifier et gérer ce changement.	9,1	12
2. Consacrer les ressources nécessaires à la compréhension du problème.	8,7	10
3. Les compromis étant inhérents au génie logiciel, il faut les rendre explicites et les documenter.	8,4	11
4. L'incertitude est inévitable en génie logiciel. Il faut identifier et gérer cette incertitude.	8,0	11

Tableau 2 Principes évalués comme étant les plus pertinents

Ce premier groupe est caractérisé par la préoccupation du changement. Le premier principe reconnaît que des changements seront nécessaires et qu'il faut mettre en place les moyens d'en minimiser l'impact : documenter les éléments les plus susceptibles de changer, prévoir des étapes dans le processus dans lesquelles des changements seront permis. Le second combine le fait que les problèmes sont complexes et souvent flous lors des premières étapes. On suggère également de chercher à identifier le plus tôt possible la catégorie de problèmes à laquelle se rapporte la situation en cause. Le troisième principe traduit l'idée que tous les éléments souhaitables ne pourront pas être traités avec autant d'efforts, et qu'il vaut mieux faire des choix explicites plutôt que de les faire en catastrophe lorsque les ressources viennent à manquer. Enfin, le quatrième est à rapprocher du premier, l'incertitude étant porteuse de la nécessité du changement.

---

7 Seulement 12 des 14 répondants étaient disponibles pour la Ronde 3.

8 La formulation originale des principes en anglais se trouve à http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html.

Principes évalués comme peu pertinents

PRINCIPE	*PERTINENCE*	*NB. DE PARTICIPANTS D'ACCORD (MAX = 12)*
5. Les exigences doivent être fermes et fixes.	3,3	9
6. Les outils, méthodes et infrastructures doivent être conçus et choisis pour supporter l'ingénieur logiciel.	4,2	10
7. Traiter les différents aspects individuels d'un problème en se concentrant sur chacun d'eux séparément.	4,9	6

Tableau 3 Principes évalués comme étant peu pertinents

Les principes fondamentaux du Tableau 3 ont donc reçu les évaluations les plus faibles, ce qui veut dire que plusieurs participants les ont évalués peu pertinents. On notera qu'il y consensus sur le score moyen que sur un seul de ces principes, le sixième. Cela dénote un certain niveau de désaccord dans ces cas.

Le cas du cinquième principe est intéressant parce que celui qui l'a suggéré a aussi soumis un principe de la catégorie précédente. Dans l'esprit de ce participant, la coexistence des deux réalités constitue le paradoxe fondamental du génie logiciel : la nécessité d'en venir, au moins à l'étape de la programmation, à fixer les exigences alors que la réalité par laquelle et pour laquelle elles sont produites change constamment. Nos participants ont cependant fortement privilégié la prise en compte du changement. Un d'eux a d'ailleurs écrit que les exigences ne sont jamais fixes plus d'une journée.

Les promoteurs du sixième principe arguaient que l'individu faisait la différence entre un produit de qualité et un autre qui ne le serait pas, et qu'en conséquence il fallait organiser le travail et les outils en fonction de la performance individuelle. Les opposants disaient plutôt que l'individu n'est pas plus pertinent que le client, et que le processus est l'élément central qu'on doit favoriser.

Enfin, alors que certains prétendaient que le septième principe offre la seule façon de maîtriser les projets complexes, d'autres l'ont rejeté en disant qu'il s'agissait justement là de l'erreur principale du génie logiciel, c'est-à-dire de découper le problème en «solutions» trop petites pour être finalement intégrées correctement. Le principe est donc peut-être acceptable, mais il n'a pas été compris de la même façon par tous. Commentaire intéressant d'un participant d'après lequel il s'agirait là d'un domaine où le génie logiciel peut contribuer à la science en général, par sa façon d'aborder les situations complexes.

Quelques surprises?

PRINCIPE	*PERTINENCE*	*NB DE PARTICIPANTS D'ACCORD (MAX = 12)*
8. Appliquer et utiliser des mesures quantitatives dans la prise de décision.	7,6	9
9. Implanter un processus rigoureux et l'améliorer continuellement.	6,9	7
10. Définir les artefacts logiciels de façon rigoureuse.	6,4	8
11. Gérer d'une manière aussi formelle que possible la qualité durant l'ensemble du cycle de vie.	7,8	9

Tableau 4 Quelques surprises !

Les principes présentés au Tableau 4 sont intéressants parce que les résultats qu'on y trouve sont parfois surprenants. Plusieurs associent le génie logiciel avec mesures, discipline et rigueur. Pour eux, les résultats du Tableau 4 peuvent être déconcertants. L'examen des évaluations individuelles nous apprend cependant que deux groupes se distinguent : ceux qui préconisent cette approche quantitative, et ceux qui y croient moins. Les résultats ci-dessus représentent une moyenne entre deux pôles. Les étapes ultérieures de nos travaux nous éclaireront là-dessus : ces résultats proviennent-ils de la nature de l'échantillon, ou bien la communauté leur accorde-t-elle réellement une pertinence moyenne?

Le huitième principe ci-dessus a été précisé ainsi : mesurer ce qui est nécessaire et utile seulement, et qu'il ne faut pas forcément des mesures quantitatives. On signale aussi que cela peut avoir des effets pervers : données fausses, distorsion du processus dans le sens des éléments mesurés au détriment d'une vision plus large, etc.

Le numéro neuf est très proche de précédent. Alors que certains disaient qu'il s'agit d'un principe des plus fondamentaux, d'autres en voyaient plutôt les dangers sous-jacents : une discipline imposée est plus nuisible qu'utile.

Le dixième a causé aussi quelques interrogations, surtout à propos du terme «rigoureusement». Quelqu'un a suggéré plutôt «qualité» alors qu'un autre a fait la mise en garde à l'effet que, si rigueur et précision convenaient, un formalisme trop lourd était dangereux. D'autres encore soulignent que, comme dans le cas de la qualité, le dosage adéquat est la meilleure solution.

Enfin, le dernier principe de ce groupe a fait ressortir quant à lui l'écart entre ceux qui veulent encore plus de formalisme et de rigueur et ceux qui semblent croire que c'est impossible.

Les autres...

PRINCIPE	*PERTINENCE*	*NB DE PARTICIPANTS D'ACCORD (MAX = 12)*
12. Construire en réutilisant et pour être réutilisé.	8,0	9
13. Mettre en place un processus logiciel flexible.	7,6	10
14. Minimiser les interactions entre les composants logiciels.	7,3	11
15. Produire le logiciel par étapes.	7,7	8
16. Établir des objectifs de qualité pour chaque produit à livrer.	7,7	11

Tableau 5 Autres principes suggérés

Bien qu'ils ne ressortent pas de l'ensemble, ces principes restent intéressants. On peut souligner d'ailleurs leur position relativement au groupe des principes évalués comme étant les plus pertinents.

Le douzième est un sujet très discuté notamment dans le cadre de l'orientation vers l'objet. Quelqu'un a posé la question : comment se fait-il que nous n'ayons pas plus de pièces interchangeables?

Compte tenu de l'importance accordée au changement, on pourrait croire que la flexibilité soit aussi jugée pertinente. Cependant, ce ne fut pas le cas et nous n'avons pas eu de commentaires assez précis pour interpréter l'évaluation du treizième principe.

Le quatorzième semble présent depuis des décennies dans le domaine et est souvent exposé dans les manuels comme étant le besoin de minimiser le couplage.

Le quinzième représente aussi une pratique très connue. Cependant, les experts ont interprété ce principe de deux façons : s'agit-il de produire des versions successives chacune plus complexe, ou de produire des versions chaque fois plus précise, plus près des exigences. Un répondant a tout de même ajouté qu'il s'agit de la pratique la plus fondamentale que nous ayons acquise depuis 20 ans.

Enfin, le dernier principe se rapproche de ceux sur la rigueur des artefacts et sur la gestion de la qualité dans l'ensemble du processus : les répondants soulignent la nécessité d'adapter le niveau de qualité selon l'élément en cause, et qu'il ne faut pas viser le même niveau pour tout.

Conclusions

Cette étude a été suivie par un atelier de travail dans le cadre d'ISESS'97[9], où une vingtaine de participants ont émis leur opinion sur les résultats obtenus. De cet atelier, il ressort une liste de critères améliorés, ainsi que des formulations plus précises pour quelques-uns des principes présentés ici. Ces améliorations seront utilisées pour les étapes subséquentes.

Il ressort quelques conclusions plus générales sur les réponses du groupe Delphi. On constate tout d'abord que le changement a été une préoccupation très forte dans ce groupe et qu'elle l'a emporté sur les considérations de discipline, de mesure et de contrôle.

Seconde constatation, le panel est divisé en ce qui concerne le type de mesures et de contrôle qui doit être exercé : certains préconisent le quantitatif, mais pas tous. L'atelier de ISESS'97 a immédiatement fait valoir que le quantitatif faisait partie des fondements incontournables du génie, et qu'on ne pouvait remettre en question l'importance de cette approche si on doit parler de «génie» logiciel.

Enfin, dernière constante dans les commentaires, la préoccupation pour le dosage approprié de discipline, de recherche de qualité et de mesure. Cela pose problème dans le sens où l'identification de ce qui est le niveau «approprié» est très difficile à uniformiser.

Cette première consultation, quoique riche, comporte évidemment des limites dont notamment le nombre limité de participants et le fait que la synthèse de leurs suggestions aurait pu être réalisée autrement et donner une autre liste. En conséquence, cette étape sera suivie d'une seconde étude Delphi auprès d'un groupe d'officiels en génie logiciel de l'IEEE Computer Society. Il s'agit de consulter cette fois un groupe plus homogène et plus représentatif des utilisateurs d'abord visés pour les principes fondamentaux.

Cette seconde étude produira une liste plus raffinée de principes fondamentaux, liste qui sera soumise ensuite à l'ensemble des membres de la *Computer Society* de l'IEEE. Nous voulons ainsi confronter les principes fondamentaux à l'opinion des praticiens.

D'autres activités parallèles sont envisagées : nous comptons analyser le corpus de normes actuel, l'ensemble des connaissances du domaine, telles que codifiées dans les manuels courants, ainsi

---

[9] ISESS'97, Third International Symposium and Forum Software Engineering Standards, Walnut Creek, CA, IEEE Computer Society, juin 1997.

que les programmes universitaires dans le domaine du génie logiciel afin d'évaluer la couverture qu'on y trouve des principes fondamentaux. Ainsi, nous espérons identifier des failles dans l'un ou l'autre de ces éléments : des principes fondamentaux dont on ne tient pas compte ou qui n'ont pas une place cohérente avec les résultats de nos enquêtes.

Rappelons qu'un des déclencheurs de ces travaux était le manque de support empirique pour certaines parties des normes promulguées. Il s'ensuit donc que divers travaux expérimentaux devraient avoir lieu afin de confronter les principes fondamentaux aux pratiques réelles dans le but de tenter d'observer leur effet réel en contexte de développement.

Bibliographie

Boehm, B. W., 1983, «Seven Basic Principles of Software Engineering», *The Journal of Systems and Software*, Vol. 3, No. 1, pp. 3-24, Mars.

Brooks A., J. Daly, J. Miller, M. Roper et M. Wood, 1995, *Replication of experimental results in software engineering*, Working Paper EFoCS-17-95, Strathclyde University Computer Society Department, Disponible à http://www.cs.strath.ac.uk/CS/research/efocs/abstracts.html#EFoCS-17-95.

Brooks F.P. Jr, 1996, *Le mythe du mois-homme*, International Thomson Publishing France, 2e éd.

Coallier F. et P.N. Robillard, «An Overview of Software Engineering Standards Activities in the IEEE», *Proceedings of CIPS Session 85, pp.134-138.*

Glass R.L., 1996 «The Relationship Between Theory and Practice in Software Engineering», *Communications of the ACM*, Vol. 39, no11, Novembre, pp. 11-13.

Jabir, 1997, «A Search For Fundamental Principles of Software Engineering, Report of a Workshop conducted at the Forum on Software Engineering Standards Issues» (1996), Sera publié dans *Computer Standards & Interfaces - The International Journal on the Development and Application of Standards for Computers, Data Communications and Interfaces*, North-Holland, Elsevier Science. (Les participants à cet atelier se sont donnés le nom collectif de Jabir) Également disponible à http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

McConnell S., 1996, «Who Cares About Software Construction?», *IEEE Software*, Vol. 13, No. 1, Janvier, pp. 128-129.

McConnell S., 1997, «Software's Ten Essentials», *IEEE Software*, Mars/Avril, Vol 14, No. 2 pp. 143-144.

Moore, J. W., 1997, *Software Engineering Standards - A User's Road Map*, IEEE Computer Society Press, Los Alamitos, Californie, 328 p.

Parnas, D.L., 1995, «Software Engineering : An Unconsummated Marriage», Conférence prononcée à : *Software Engineering Symposium (ISESS'95)*, Montréal, 21-25 août.

Parnas, D.L., 1997, «Software Engineering : An Unconsummated Marriage», *Communications of the ACM*, Vol. 40, no. 9, Septembre, p.128.

Ramamoorthy C.V. et W. Tsai, 1996, «Advances in Software Engineering», *IEEE Computer*, Vol. 29, No. 10, Octobre, pp. 47-58.

Read W.S. et J. Iorio, 1994, «Streamlining the Standards Development Process», *IEEE Canadian Review*, Printemps-été, pp 13-14.

Robinson G.S. et C. Cargill, 1996, «History and Impact of Computer Standards», *IEEE Computer*, Vol. 29, No. 10, Octobre, pp. 79-85.

Shaw M., 1994, «Prospects for an Engineering Discipline of Software», *Encyclopedia of Software Engineering*, John Wiley, Vol. 2, 1994, pp. 930-940.

Tripp L. et P. Voldner, 1995, «A Market-Driven Architecture For Software Engineering Standards», *Proceedings of the Second IEEE International Software Engineering Standards Symposium (ISESS'95)*, IEEE Computer Society, pp. 105-116.

Remerciements

Nous tenons enfin à remercier toutes les personnes dont les idées se retrouvent d'une façon ou l'autre dans ce travail : les répondants de l'étude Delphi ainsi que les participants aux ateliers de travail de Montréal et de Walnut Creek. Les auteurs désirent également remercier M. Leonard Tripp, IEEE Software Engineering Standards Committee Chair, pour son soutien à cette démarche. Le Laboratoire de recherche en gestion des logiciels de l'Université du Québec à Montréal est en partenariat avec Bell Canada avec du soutien financier additionnel du Conseil de recherches en sciences naturelles et en génie du Canada.