Génie logiciel : canaliser la créativité

DÉBAT RICS ORGANISÉ À LA FNAC LE 23 MARS 1989

Dans la production du logiciel, les méthodes se substituent au bricolage, plus structurées, plus liées à ces boîtes à outils que sont les « ateliers de génie logiciel » ; on en attend productivité mais aussi augmentation de la qualité. Plusieurs questions se posent : l'évolution des professions concernées (normes de travail, rapport au produit, formation, etc.), et la coexistence de plusieurs marchés du travail.

Pierre Salkazanov : Au département des études de Bull, je suis responsable des méthodes de développement. Ces dernières années, on a fait beaucoup plus de progrès dans le développement du matériel que dans le domaine du logiciel. Chez Bull plus de 1 500 personnes sont concernées par des méthodes communes, pour maîtriser le processus de développement ; mais en même temps ces personnes ont envie d'avoir le plus de liberté possible, d'utiliser le moins de méthodes possible. Il faut trouver un compromis entre globalisation et différenciation.

Jean-René Abrial : Ecrire un programme est une activité étrange qui consiste à commander une machine à distance dans le temps, à se mettre dans l'idée du fonctionnement de cette machine alors qu'elle n'est pas en activité, à dire comment la machine va fonctionner beaucoup plus tard. Dans l'histoire des techniques, c'est nouveau : dans d'autres disciplines on fabrique des appareils, on construit des routes, des maisons ; ici nous télé...conduisons dans le temps.

C'est une activité qui n'a pas de tradition. Depuis 40 ans, on cherche désespérément à apprendre, on arrive à trouver des méthodes et des outils... Mais ceux-ci en sont à leurs premiers balbutiements.

Michel Crampes : Dans les sociétés de service, nous fournissons les clients en essayant de ne pas dépasser nos budgets. Un client a un besoin, nous obtenons un marché et nous mettons en place une équipe d'ingénieurs qui vont essayer de comprendre, de dessiner une solution, de faire des plans, de fabriquer la solution, puis de la tester. Le besoin d'un système informatique est éminemment flou, mou, malléable. L'équipe d'informaticiens va durcir ce besoin, l'expliciter, le clarifier. Les méthodes et les outils sont là pour leur permettre de travailler de manière raisonnée.

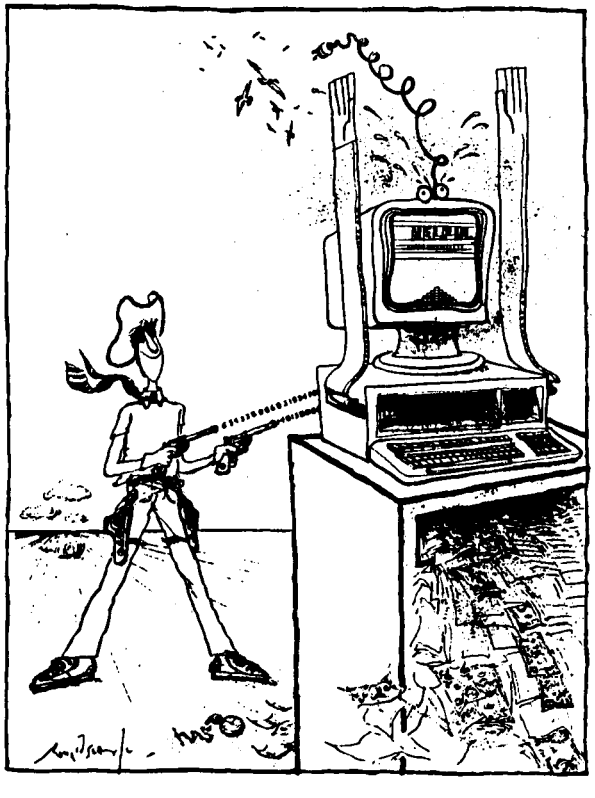
François-Yves Vuillemin : L'informatique est pleine de méthodes et d'heuristiques [1]. Elles viennent

1 *Programme non déterministe centré vers la recherche.*

de « trucs » qu'ont trouvés les programmeurs pour se tromper le moins possible, pour transformer la spécification (le besoin dur) en une ligne de code, écrite dans un langage de programmation. On cherche à automatiser, à fournir des outils aux programmeurs les aidant à produire leur code.

Jean Lojkine : On n'automatise pas un geste physique, une fonction opératoire, mais une fonction cérébrale et pourtant on continue à vouloir évaluer la productivité de ce travail, son coût, dans les termes d'une culture fondée sur des produits matériels. Les ingénieurs logiciel ont un rapport spécifique à leur travail (bricolage, fonctions imaginatives, créativité) qui ne recoupe pas les préoccupations des autres fonctions de l'entreprise. Leurs problèmes sont des problèmes d'écrivains. On parle d'écriture à propos de la programmation. On peut aussi mettre l'accent sur le caractère routinier, enfermant et taylorisant des tâches des jeunes programmeurs ; avec cependant certains instants, en particulier en début de projet, d'une grande richesse, d'une grande créativité. La complexité de leur travail demande une formation plus importante, mais cela bouleverserait les formations autodidactes qui prévalent jusqu'ici dans l'informatique.

Pierre Salkazanov : Le travail de conception de logiciel est présenté comme créatif et humain, laissant les programmeurs « propriétaires de leur code » : une fois écrit personne ne peut plus le comprendre. Mais, aujourd'hui, des possibilités de maîtriser la « production du logiciel » comme celle du matériel existent. Le logiciel a accumulé un retard dans les méthodes parce qu'on le considérait comme mou, facilement modifiable, au contraire du matériel. Ainsi, lorsqu'on fait un circuit VLSI [2] de 500 000 transistors et qu'il ne fonctionne pas, cela coûte trois ou quatre mois pour le recommencer ; donc un retard équivalent pour le projet. Pour maîtriser la conception des VLSI, on a donc inventé des méthodes de conception et de simulation, introduit de la rigueur dans les développements. Les ingénieurs l'ont admis : s'il restait une erreur, le projet risquait de se planter.



En logiciel, en cas d'erreur, on pensait la corriger facilement : donc pas la peine de faire trop d'efforts en amont... Mais quand 100, 200, 1 000 ingénieurs de développement font ce raisonnement, il n'y a pas une erreur mais 1 000, 5 000... et là les conséquences sont très importantes...

Terminal : *Les exigences de qualité et d'outillage sont-elles les mêmes pour les logiciels de base [3] et pour les logiciels d'application ?*

François-Yves Vuillemin : Un programme doit être livré correct, mais il est difficile de faire la « preuve » de ce programme, de vérifier que le code obtenu est conforme aux spécifications. Au sein de l'automatisation des activités intellectuelles de l'homme, de ses activités de création, le génie logiciel n'est qu'une toute petite partie. Herbert Simon [4] pose la question de savoir si le génie logiciel doit être « intelligemment artificiel ». La part créative de l'activité humaine va plus passer du côté des outils et des méthodes, de façon à ramener le travail du programmeur à un travail de compréhension des besoins de l'utilisateur, puis de production de code.

Terminal : *Les erreurs dans la conception du logiciel relèvent-elles uniquement de facteurs techniques, des mathématiques formelles, ou bien ne faut-il pas prendre en compte un autre plan, celui du sens, de la responsabilité ?*

Pierre Salkazanov : Une première partie du travail consiste à comprendre les besoins de l'utilisateur, à les formaliser ; ensuite il faut faire l'architecture du produit, l'élaboration des spécifications fonctionnelles, la conception de haut niveau, la conception de bas niveau, puis la programmation. Au lieu de se précipiter pour écrire le code, on prend du temps pour concevoir en s'aidant d'outils graphiques et des langages, en essayant d'apporter des preuves et des simulations. Au bout du compte, l'activité de programmation, proprement dite, devient toute petite – moins de 10% – par rapport au cycle complet de développement du logiciel. On a donc plus besoin d'ingénieurs de haut niveau, capables de maîtriser l'ensemble du processus, que d'un découpage entre concepteurs, analystes et puis, derrière, les programmeurs venant écrire le code.

Terminal : *Que pensez-vous de l'utilisation de langages formels dans la phase de spécification ?*

Pierre Salkazanov : Pour maîtriser la complexité du problème on le décompose en un certain nombre de modules et on s'efforce d'obtenir une organisation hiérarchisée. On l'exprime globalement, puis on l'affine de plus en plus. Ainsi, on arrive à décrire le produit logiciel avec un arbre en terme de « composants » et d'interfaces entre les composants. La conception sera donc un travail plus intéressant que par le passé.

Michel Crampes : Le logiciel est un produit transparent, il n'obéit pas à grand chose, il est flou, il peut se promener, c'est un automate, une machine. Un premier syndrome illustre cette « invisibilité » : celui du producteur, qui, interrogé à un quelconque moment sur l'état d'avancement de son projet, répond le plus souvent qu'il est fait « à 90% ». La première réalité

2 *VLSI : very large scale integration.*

3 *Logiciel de base : première couche de logiciel permettant d'utiliser le matériel (système d'exploitation, ex. MSDOS).*

4 *Herbert Simon, spécialiste américain de l'intelligence artificielle.*

c'est donc d'abord la programmation ! Le caractère spécifique de cette activité vient de ce que chacun des programmes est conçu comme un objet unique ; comme si lorsqu'on construisait une maison, à chaque brique on réinventait la brique, chacune d'entre elles ayant une forme ou une taille particulière. Actuellement dans le monde, des milliers d'ingénieurs fabriquent différemment les mêmes programmes et s'ignorent.

Le second syndrome est celui de « Lucky Luke » : l'aura médiatique de l'ingénieur est telle que, pour conserver son prestige, il doit programmer quasiment plus vite que la demande qui lui est faite. Il « dégaine » donc sur le clavier vraiment très rapidement.

Or, le problème des professionnels du logiciel n'est pas de tirer à vue, mais de fabriquer des choses répondant à un besoin marchand, donc de définir des métiers et des structures de production. Pour produire un logiciel complexe, on fabrique une structure de production appelée « groupe projet » : une équipe va naître, s'organiser puis mourir. Le chef de projet constitue son groupe de jeunes ingénieurs ; il les conduit à plus de méthode et les incite à communiquer entre eux, et pas seulement avec la machine, afin de produire du logiciel qui réponde aux spécifications et pas une œuvre d'art liée à son créateur.

Pierre Lévy : La programmation est un genre d'écriture, à la fois très nouveau et très ancien, l'écriture ayant toujours été une façon de télécommander une lecture à distance dans le temps. Comment automatiser l'écriture ? Difficile de répondre, puisqu'il n'y a pas de méthode pour écrire un bon texte. Cela dit, selon la nature des textes – lettre administrative ou lettre d'amour très inspirée – il y a des genres d'écriture automatisables qui ont un aspect standard, d'autres pour lesquels ce n'est pas vrai. La programmation est une écriture très particulière, une traduction en somme : elle exprime les besoins de l'utilisateur et les traduit dans un langage avec des contraintes formelles extrêmement fortes. Cette écriture sous contrainte est également une activité de communication, non seulement avec les hommes mais aussi avec les machines.

Jean-René Abrial : Une activité d'écriture, certes, mais qui, vu la taille et la complexité des problèmes, ne peut plus être individuelle : il s'agit d'écrire à plusieurs. Ensuite dans l'idée d'écrire une lettre, il y a une idée d'unité de temps ; alors que dans la construction du logiciel, vu le caractère fluctuant des besoins, cette écriture à plusieurs dure longtemps. De plus, la plupart des programmes ne sont pas du tout en contact avec des utilisateurs, mais avec des appareils : le programme embarqué d'Airbus qui fait atterrir l'avion remplace le pilote. Ce qui pose la question de la sécurité des programmes. Les programmes sont des objets très fragiles : une toute petite chose fausse peut déclencher une erreur catastrophique. Dans la création des artefacts, la sensibilité, la fragilité de ce qu'on fabrique constituent quelque chose d'entièrement nouveau.

Jean-René Abrial : L'enjeu c'est de faire un travail collectif complexe et de ne pas laisser les informaticiens enfermés dans leur bulle.

Jean Lojkine : Peut-on concevoir des systèmes de production déhiérarchisés où il n'y aurait pas seulement les spécialistes mais aussi les usagers des systèmes informatiques, de façon à créer un rapport spécifique entre l'homme et la machine informationnelle ? Il est difficile de parler de métro sans conducteur ou d'avion sans pilote : l'homme est toujours quelque part. Le dialogue entre l'homme et la machine est fondamental, ce qui oblige à révolutionner nos habitudes culturelles issues de la révolution industrielle. ■

Les logiciels Hypertexte

« Nous vivons dans un monde de papier. Mais en fait lorsqu'il s'agit de représenter la pensée, le papier s'avère une solution médiocre : les idées ne procèdent pas en séquence. Lorsque nous écrivons, nous sommes obligés de prendre ces pensées organisées de façon complexe, et de les arranger les unes à la suite des autres. L'hypertexte permet de s'échapper de cette notion de séquence et de présenter les idées tout en tenant compte de leur interconnexion. Il permet de mieux faire apparaître leur véritable structure. Les lecteurs peuvent en tirer une compréhension plus riche des matériaux étudiés » écrit en 1970 Ted Nelson, un des pionniers de l'hypertexte dans *Computer Lib and Dream Machines (Bibliothèque électronique et machine de rêves)*, un livre-manifeste. Ses idées attendront vingt ans pour être reconnues. Le disque optique numérique offre aujourd'hui la possibilité de stocker une salle entière dédiée aux archives en quelques disques seulement. Il reste un problème : comment retrouver une information précise dans cette immense réserve ? Les bases de données ont longtemps été considérées comme la solution. Mais il leur manque généralement la possibilité de créer des liens entre les documents de natures diverses : textes, images, graphiques. C'est ce que tentent d'offrir les logiciels de type hypertexte qui permettent une organisation et une restitution plus aisées de l'information. L'utilisateur d'un tel logiciel pourra suivre le cours de ses idées, cliquer ou ne pas cliquer sur un bouton pour parcourir la documentation ou le dossier qu'il consulte.
Des logiciels possédant certaines fonctionnalités hypertexte existent déjà sur micro-ordinateurs. Le plus connu d'entre eux est Hypercard, le logiciel qui accompagne le Macintosh.

Propos mis en forme par Eric Braine.
Le débat était animé par notre ami Jean-Pierre Cahier.

Pierre Salkazanov (Bull), Michel Crampes (Syseca), François-Yves Vuillemin (CNAM), Jean Lojkine (CNRS), Jean-René Abrial (consultant indépendant), Pierre Lévy (Terminal).