01 informatique mensuel

N° 149 avril 1981

couverture : YOR

Puissance et intelligence, performances et subtilité... aucune impasse n'est envisageable : le progrès de l'informatique sera, à la fois, qualitatif et quantitatif ou ne sera pas.

Les 32 bits ? Apparemment une simple question de puissance. La rapidité de l'évolution technologique permet ainsi de doper des systèmes que l'habitude seule autorise encore à désigner du nom de « minis ». Un nouveau pas vient d'être franchi avec le premier microprocesseur à 32 bits d'Intel. Que faut-il admirer davantage ? La puissance disponible sous ses trois minuscules boîtiers (approximativement celle d'un 370-148) ou l'intelligence d'une micro-machine capable de parler Ada ?

La gestion documentaire ? D'abord un problème de volume : le développement de la micrographie et, bientôt, celui du disque optique numérique devraient reléguer définitivement au rang des souvenirs cet océan de papier que la vague informatique n'a cessé d'alimenter. Supposons le problème résolu et l'information stockée, au moindre coût, dans un récipient d'accès commode et rapide. Encore faudra-t-il disposer de moyens efficaces de recherche, de sélection, de synthèse, de choix. Les bases de données documentaires qui apparaissent sur le marché tracent la voie d'une informatique davantage au service de l'homme.

L'augmentation exponentielle de la puissance disponible et des volumes à traiter ne permettra plus l'imprécision ni l'à-peu-près, qui ont trop souvent été le lot de nombreux projets logiciels. Performant et subtil, le système informatique requiert des outils logiciels parfaitement au point. Il requiert surtout une rigueur absolue dans la définition des objectifs qui lui sont confiés. Entre l'homme et la machine, le dialogue doit être clair. L'expression précise du besoin apparaît, désormais, comme la phase la plus délicate dans la mise en œuvre d'une nouvelle informatique, puissante et intelligente. Chaque informaticien est concerné par les outils de spécification dont nous décrivons, dans ce numéro, le rôle et l'importance.

Bernard Sauteur

La spécification : un outil pour l'informaticien

« Ce qui se conçoit bien s'énonce clairement... ». Les langages de spécification pallient les faiblesses du langage naturel et laissent entrevoir la possibilité d'une programmation réellement automatisée.

par Michel Demuynck et Bertrand Meyer

Les dernières années ont vu une prise de conscience croissante des résultats techniques et économiques en jeu dans ce qu'on a appelé la « crise du logiciel ». Le seul chiffre de 250 milliards de francs – coût estimé de l'informatique dans le monde en 1975 —, et le fait que ces coûts sont à 75% et pour une proportion qui ne cesse de croître des coûts de logiciel, montre qu'il est crucial d'améliorer les méthodes d'étude et de maintenance des systèmes informatiques. Dans ce but, de nombreuses idées ont été proposées, qui ont nom « programmation structurée », « programmation modulaire », « programmation systématique », « analyse descendante », « analyse composite », etc. ; leurs partisans insistent sur la nécessité d'une démarche logique et rigoureuse, sur l'importance de la clarté et de la documentation des programmes, sur l'emploi de structures de contrôle et de données bien définies.

Bien que ces idées soient fructueuses et fondamentales, l'expérience montre qu'elles ne résolvent qu'une partie du problème. Lorsque en effet on en arrive à l'étape de la programmation, tout est déjà joué pour une large part : les véritables difficultés se placent avant, au moment où l'on cherche à comprendre et décomposer le problème. L'importance de cette phase est particulièrement évidente dans un domaine comme l'informatique de gestion, où la plupart des tâches à effectuer sont conceptuellement simples ; cependant, même si les éléments du système, pris individuellement, semblent « faciles », leur nombre même et leur imbrication rendent extrêmement ardue la compréhension de l'ensemble.

Savoir poser les problèmes

Une des conséquences de cette situation, qui se produit dans tous les domaines d'application, est qu'il manque la plupart du temps un bon cahier des charges. Le cahier des charges est trop souvent énorme, complexe, difficile à comprendre et difficile à contrôler.

Vérifier qu'il est cohérent et complet est une tâche impossible dans ces conditions. Le résultat le plus clair est l'apparition au stade de la programmation de difficultés et de choix qui auraient dû être réglés au cours de l'« analyse » et qui peuvent perturber gravement, voire mettre en péril, la réalisation du projet de programmation.

Le concept de langage de spécification correspond à cette recherche d'une méthode pour poser les problèmes avant de commencer à les résoudre. Il s'agit de fournir un cadre précis pour constituer sous une forme fiable et rigoureuse, pouvant éventuellement être traitée automatiquement, l'exposé d'un problème que l'on envisage de résoudre sur ordinateur. Il faut réfléchir avant d'agir, et une tâche a d'autant plus de

Cet article est inspiré du chapitre « Spécification » de l'ouvrage collectif « Manuel de génie logiciel » à paraître.

chances d'être menée à bien qu'elle a été précisément définie. On entretient plus facilement un programme bien codé ; on code mieux un programme bien conçu.

L'application de ce principe à l'ensemble du cycle de vie d'un logiciel conduit à examiner avec un soin tout particulier la première des phases proprement techniques : celle qui consiste à définir le problème à résoudre. C'est la spécification.

Spécifier un système, c'est définir de façon aussi complète et peu ambiguë que possible les caractéristiques externes qu'il doit présenter à ses utilisateurs potentiels ou au système dans lequel il s'insère.

Un impératif méthodologique

Le terme de spécification est passé depuis relativement peu de temps au premier rang des mots vedettes en génie logiciel. L'importance de l'activité qu'il recouvre a cependant été reconnue depuis longtemps, en particulier dans le domaine de l'informatique de gestion où l'on a plutôt coutume de parler d'analyse fonctionnelle. Telle qu'elle est pratiquée, cette dernière tâche recouvre en partie des activités de conception, c'est-à-dire de résolution, même très générale, des problèmes ; mais par d'autres aspects elle se rattache en grande partie à la spécification, c'est-à-dire à la définition de ces problèmes.

Le regain d'attention dont bénéficie actuellement la spécification résulte d'une confluence entre l'évolution des travaux sur la méthodologie de la programmation et les problèmes concrets grandissants rencontrés par les industriels dans la maîtrise des projets qu'ils développent.

Le premier phénomène est lié à une meilleure compréhension, et à une certaine remise en cause des idées sur la démonstration de validité des programmes. Pour démontrer la validité d'un programme, il faut commencer par exprimer quel est son comportement souhaité, quel problème il est censé résoudre. Tant que la recherche en ce domaine est restée confinée à des exemples d'école (calcul d'un PGCD, d'une factorielle...), l'étape de définition des objectifs n'a pas semblé particulièrement difficile. Il en est allé tout autrement lorsque, les mécanismes techniques fondamentaux de la démonstration ayant été élucidés, on a voulu s'attaquer à des problèmes plus ambitieux. L'expression même de ce qu'il fallait démontrer, c'est-à-dire de ce que les programmes étaient censés calculer, est apparue alors comme l'une des tâches les plus délicates. On a pu s'apercevoir en revanche dans certains cas que lorsque cette difficulté était surmontée, et que l'on disposait d'une description précise du problème à résoudre, la résolution elle-même, c'est-à-dire l'écriture du programme, pouvait être conduite de façon systématique, voire automatisée en tout ou en partie. C'est le vieux rêve de la synthèse de programmes, domaine de recherche qui connaît aujourd'hui un regain d'activité.

Déceler les erreurs au plus vite

L'autre raison de l'intérêt actuellement porté aux spécifications est pratique, et nous importe plus ici : c'est la constatation qu'un grand nombre d'échecs de projets logiciels et d'imperfections dans des programmes menés à leur terme sont dûs à des erreurs commises à la base, au moment de la définition des objectifs : spécifications incomplètes, incohérentes, impossibles à satisfaire, ambiguës, ou tout simplement inexistantes.

Il a fallu attendre un certain « effet de taille » pour que le problème se manifeste dans toute son ampleur. C'est ainsi qu'une étude de Bell et Thayer montre que, dans une spécification de 2 500 pages et 8 248 paragraphes, 972 erreurs avaient été décelées. Pour qui a une certaine habitude de l'examen attentif des spécifications, ce nombre semble d'ailleurs indiquer que l'exemple étudié n'était pas, et de loin, des plus mauvais.

L'une des principales raisons d'être des spécifications apparaît, à la lumière de ce qui précède, comme directement liée au phénomène des erreurs : il s'agit de trouver les erreurs aussi tôt que possible dans le cycle de vie et de circonscrire les erreurs ultérieures au domaine des détails faciles à corriger. Bien que les chiffres précis diffèrent d'un auteur à l'autre, il est clair que le coût de la correction d'une erreur de spécification au stade des tests est supérieur de plusieurs ordres de grandeur à ce qu'on dépense lorsque l'erreur est détectée dès la phase de spécification.

Améliorer les relations avec les clients

Au-delà de leur intérêt strictement technique, les spécifications jouent également un rôle contractuel important. L'une des sources majeures de conflits entre clients finals et prestataires de services est la mésentente sur ce que doit faire le système réalisé ; il n'est pas rare que des malentendus fondamentaux ne soient découverts qu'au moment de la livraison du produit final, alors qu'il est trop tard, depuis longtemps, pour y remédier. Les aspects légaux liés à ces questions sont extrêmement délicats et mal définis, précisément parce que en l'absence de documents précis il est impossible de décider objectivement en quoi consistent les engagements réciproques.

L'existence d'une spécification de bonne qualité est un pas important vers l'amélioration des rapports entre les informaticiens et les destinataires du projet. Les malentendus éventuels, lorsqu'ils apparaissent noir sur blanc dans un document précis, peuvent en général être résolus simplement. Pour les discussions ultérieures avec les clients, qui devront inévitablement se poursuivre tout au long du projet, les informaticiens disposeront d'une base de référence commune, sur laquelle un accord a été établi. Lorsque le cadre légal est contraignant (pénalités en cas d'échec ou de retard, obligation de satisfaire à certains seuils de performances, etc.), le document de spécification est la seule protection du réalisateur du système contre les interprétations abusives, les exigences injustifiées, les clauses rajoutées a posteriori. Dans certaines circonstances, la conception du système ne commencera que lorsqu'un accord sans réserve aura été obtenu sur le document de spécification, chaque section étant paraphée par les deux parties, et le client s'engageant à figer ses exigences pour une période déterminée. Il est clair que l'emploi de telles procédures exige un formalisme de spécification précis et rigoureux ; cela sera repris plus loin.

Spécifier : une tâche de génie logiciel

La spécification est, dans le génie logiciel, une spécialité à part entière, justiciable de la même méthodologie et de la même problématique que les autres étapes. On rencontrera donc à propos de la spécification :

— des problèmes de documentation ;
— des techniques de contrôle de qualité, des « revues de spécification », etc. ;
— des problèmes de mise à jour et de suivi des documents ;
— des outils de gestion, de validation et de documentation des spécifications.

Ces différents points nous aideront dans la comparaison des méthodes existantes. Auparavant, il importe de préciser la notion de spécification en la comparant à celle de cahier des charges.

Le cahier des charges, point de vue de l'utilisateur

Dans la pratique industrielle, ce qui constitue la spécification est souvent un document décrivant les exigences de l'utilisateur futur ; le cahier des charges. Il s'agit d'un document rédigé en langage humain (non formel) et émanant des services du « client », intérieur ou extérieur à l'entreprise, parfois en collaboration avec le service informatique qui sera chargé de la conception du système.

Le cahier des charges est un document capital, et son importance n'apparaît jamais plus clairement que lorsqu'on a affaire à un projet pour lequel il n'existe pas de cahier des charges proprement dit, la « spécification » se limitant alors à quelques comptes rendus de réunions, à une lettre, voire à une conversation. Nous allons cependant tenter de montrer en quoi, même s'il existe et a été convenablement rédigé, un document en langage courant ne suffit pas lorsqu'on veut réaliser un logiciel de façon à en garantir la fiabilité.

Les faiblesses du langage naturel

Le langage usuel est admirablement adapté à un grand nombre d'applications, de la déclaration d'amour à la description de la mathématique formelle, sans oublier les manuels de génie logiciel. Mais il ne convient pas pour la spécification précise d'un système informatique. On peut trouver à cela les raisons suivantes :

— ambiguïté : le langage humain est soumis à l'interprétation humaine ;

– contradiction : nous sommes habitués à la tolérer comme étant à l'image de la vie ;
– incomplétude : de nombreux éléments (l'essentiel, souvent) sont abandonnés au contexte du discours, à l'univers implicite du locuteur ;
– discursivité : le discours suit des méandres et des boucles, non une ligne déductive ;
– imprécision : les termes employés ne sont compris que par rapport à un référentiel censément commun.

Toutes ces caractéristiques du langage humain — sans lesquelles l'existence serait mortellement ennuyeuse — deviennent rédhibitoires lorsqu'il s'agit de définir aussi précisément que possible un système informatisé, qui devra ultérieurement être pris en charge par un automatisme totalement implacable. On ne rappellera jamais assez à quel point l'emploi des ordinateurs exige de la précision dans le moindre détail : aucun des « abus de langage », tolérés constamment jusque dans le langage des mathématiciens, n'est acceptable dans un langage de programmation. Si une spécification laisse dans l'ombre la réponse à une question, le programme, lui, tranchera d'une façon ou d'une autre, et pas forcément de la bonne.

Les grands écueils

Une analyse détaillée d'exemples de cahiers des charges tirés de nombreux domaines d'application permet de recenser les défauts les plus courants présentés par ce genre de documents. Ils sont résumés dans la figure 1, qui vise à fournir au lecteur une « liste de contrôle » de fautes à éviter lorsqu'il rédige un cahier des charges.

Figure 1 :
Les six péchés capitaux du spécificateur ▶

bruit	élément du texte n'apportant d'information sur aucune des caractéristiques du problème (variantes : la redondance ; le repentir).
silence	caractéristique du problème à laquelle ne correspond pas d'élément du texte.
surspécification	élément du texte ne correspondant pas à une caractéristique du problème (mais à des caractéristiques d'une solution possible).
contradiction	élément du texte définissant de façon incompatible une même caractéristique du problème.
ambiguïté	élément du texte permettant de comprendre une caractéristique du problème de deux façons ou plus.
référence en avant	élément du texte utilisant des caractéristiques du problème définies plus loin dans le texte.

On notera que cette analyse dépasse notre propos immédiat : l'emploi de méthodes de spécifications plus systématiques, telles que nous les verrons ci-après, ne garantit pas qu'on évitera toujours les défauts de la figure 1 ; de fait, cette liste de contrôle se révèle également utile lorsqu'on écrit des spécifications plus rigoureuses. Mais le langage naturel, de par ses caractéristiques inhérentes, favorise la prolifération de ces défauts dans les cahiers des charges et interdit toute vérification systématique.

La spécification : outil de l'informaticien

Malgré tous leurs défauts, les cahiers des charges sont un outil précieux, surtout s'ils sont écrits avec soin, complétés par des schémas, des index, des glossaires, etc. Pour le concepteur du système, cependant, ils ne suffisent pas. C'est ici qu'intervient la spécification proprement dite, celle qui peut être la plus précieuse pour la conception et la réalisation du système : un exposé des objectifs, mis sous une forme plus précise, plus rigoureuse, plus facile à gérer et à mettre à jour, plus cohérente, plus aisément vérifiable et validable qu'un texte en français.

Une mauvaise compréhension de la différence entre « cahier des charges » et « spécification » est à l'origine de bien des malentendus. Il est vain par exemple de reprocher à un « langage de spécification », destiné à l'expression de la seconde tâche, de ne pas être à la portée de tous les utilisateurs : destinée au professionnel, la spécification n'a pas plus de raison d'être compréhensible par le profane que le diagramme électronique ne s'adresse à l'acheteur type d'un magnétophone.

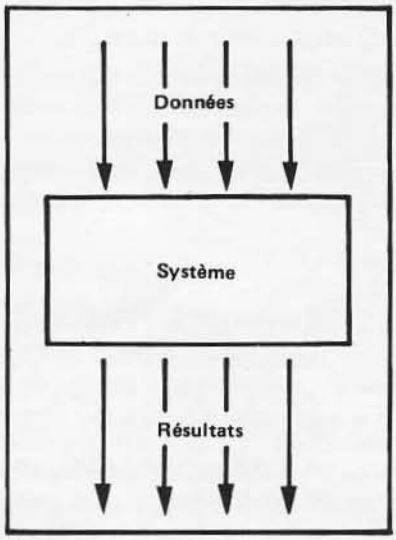

Figure 2 :
Bloc de fonction

Seule une certaine réticence à reconnaître le « professionnalisme » du travail des informaticiens peut expliquer qu'on leur dénie le droit à disposer de leurs propres outils. Certes, la situation serait idéale si l'on pouvait disposer d'un formalisme utilisable par le client aussi bien que par le programmeur ; mais cet espoir est peu réaliste, au moins à court terme.

Il convient donc de bien distinguer le cahier des charges, destiné à l'utilisateur, et la spécification, outil de l'informaticien (anglais : requirement et specification). Il est clair que le prix à payer pour la distinction entre ces deux tâches pourra être la nécessité de former un « médiateur » parlant les deux langages. Cette nécessité est particulièrement cruciale lorsque interviennent les questions contractuelles évoquées à la section précédente.

Que spécifier ?

La spécification complète d'un système informatique comprend deux parties : la spécification fonctionnelle, la spécification des performances. Ces deux tâches relèvent de méthodes très différentes et sont souvent séparées en pratique.

La spécification fonctionnelle a pour objet de décrire les caractéristiques externes du système considéré comme transformateur d'informations : ses données possibles, les résultats attendus et la relation entre les unes et les autres (souvent une fonction, d'où le nom).

La spécification fonctionnelle, ainsi définie, est une représentation d'un système, ou d'un élément de système, comme une « boîte noire » ou un « bloc de fonction » (figure 2). Elle exclut toute description des méthodes de mise en œuvre interne, des structures de données et, plus généralement, tout détail technique. En fait, une spécification fonctionnelle authentique n'indique même pas si l'élément spécifié doit être réalisé de façon logicielle (programmé) ou matérielle (câblé). Elle doit en outre rester statique, c'est-à-dire ne pas comporter d'éléments « algorithmiques » ou « procéduraux » décrivant des tâches et leur enchaînement dans le temps. C'est ce qui distingue essentiellement la spécification fonctionnelle de l'étape suivante du cycle de vie, la conception, dont l'objet est de décrire, quoique à un niveau encore abstrait, des algorithmes et des structures de données.

Bibliographie

J.R. Abrial, S.A. Schuman, B. Meyer : Specification Language ; draft, August 1979 (version française à paraître).

T.E. Bell, D.C. Bixler, M.E. Dyer : An Extendable Approach to Computer-Aded Software Requirements Engineering : IEE trans. on Software Engineering, Vol SE-3, n° 1, January 1977.

R.M. Burstall, J.A. Goguen : Putting Theories Together to Make Specifications : invited pages, Proc. 5th Intle Joint Conf. on Artificial Intelligence, Tbilissi, 1977.

M. Demuynck, B. Meyer : Les Langages de spécification ; EDF, Bulletin de la direction des études et recherches, n° 1 (1979), pages 39-60. Egalement dans : IRIA, journées sur le génie logiciel, Pont-à-Mousson, mars 1980.

R. Dewar, J. Schwartz : Abstracto, Project for an Algorithm Specification Language ; Courant Institute of Computer Science, décembre 1977.

V. Donzeau-Gouge, G. Huet, G. Kahn, B. Lang : Introduction au système Mentor et à ses applications ; BIGRE n° 15 (IRIA-IRISA), juin 1979 ; également dans : Journées francophones sur la certification du logiciel, Genève, janvier 1979

Isdos project : Problem Statement Language (PSL) - User's Manual ; University of Michigan, mars 1975.

B. Meyer : Sur le formalisme dans les spécifications ; note atelier logiciel 23, HI 3206/01.

B. Meyer, C. Baudoin : Méthodes de programmation ; Eyrolles, Paris, 1978.

D.T. Ross et al. : Special Section on Requirements Analysis ; IEEE Transactions on Software Engineering, vol. SE-3, n° 1, pages 2-34, janvier 1977.

O. Roubine, L. Robinson : Special Reference Manual ; SRI International, Technical Report CSG-45, janvier 1977.

La définition précédente de la spécification fonctionnelle s'applique immédiatement aux systèmes séquentiels ; elle se généralise au cas de systèmes parallèles ou répétitifs dans lesquels intervient un ordre temporel (systèmes d'exploitation, commande de processus, temps réel, etc.). Les méthodes utilisées pour spécifier ce type d'application se répartissent pratiquement en deux catégories :

– on peut se ramener au cas d'une relation ou d'une fonction en considérant comme entrée la suite complète (d_1, d_2,... d_m) des données fournies au système pendant une certaine période, et comme sortie la suite (r_1, r_2,... r_n) des résultats correspondants ;

– on peut élargir le modèle « relationnel » ou « fonctionnel » par des notions reflétant le caractère dynamique de l'évolution du système, en introduisant des « états » et des « événements » qui déclenchent des transitions d'état à état.

La seconde solution permet de modéliser plus étroitement le fonctionnement instantané du système, mais présente l'inconvénient de s'écarter de la notion mathématique de fonction et de perdre en partie le caractère statique requis des spécifications.

Les méthodes et langages de spécification qui seront décrits dans le prochain article couvrent essentiellement la partie fonctionnelle de la spécification.

Quant à la spécification des performances, elle décrit les conditions requises du système relativement à son utilisation des ressources physiques disponibles, en fixant des limites à ne pas dépasser. Ces contraintes sont définies, selon les cas, relativement aux valeurs moyennes en fonctionnement normal ou aux valeurs extrêmes dans les conditions favorables.

Dans le prochain numéro de « 01 Mensuel », nous passerons en revue les méthodes de nature à garantir une bonne spécification fonctionnelle. Les systèmes de spécification actuellement opérationnels seront alors décrits et critiqués.

Michel Demuynck
et Bertrand Meyer

Michel DEMUYNCK
Ingénieur chercheur au service IMA (« Informatique et Mathématiques appliquées ») de la Direction des études et recherches d'EDF.
Domaines d'intérêt : bases de données, génie logiciel.

Bertrand MEYER
Chef de division au service IMA
Animateur du groupe « Génie logiciel » de l'AFCET.
Auteur avec C. BAUDOIN du livre « Méthodes de programmation » (Eyrolles). Domaines d'intérêt : la programmation (langages, techniques, outils, méthodologie), le génie logiciel, la spécification.