01 informatique mensuel

N° 150 mai 1981

couverture : S. Duroy / Rapho

Télématique, bureautique, vidéotique, médiatique... Ils sonnent clair et moderne ces néologismes récemment apparus dans notre vocabulaire et ils donnent aux réalités qu'ils désignent un petit air familier et rassurant : comme s'il avait suffi de leur trouver un nom pour que l'essentiel soit fait. Une récente étude de l'institut Remy Genton révélait pourtant que 95 % des supports d'information actuellement utilisés sont des supports papier... Après avoir si bien décrit la bureautique et le bureau du futur, il reste apparemment à passer aux actes. Non pas que rien n'ait été fait jusqu'à ce jour. Les systèmes de traitement de textes se font de plus en plus nombreux et quelques exemplaires de bureaux automatisés ont vu le jour. Au-delà des mots, les problèmes techniques, financiers et humains doivent être résolus pour que la bureautique (l'automatisation du bureau, disent les anglo-saxons) devienne une réalité.

Les informaticiens sont concernés au premier chef par ces développements : il s'agit bien de leur problème et, pour mettre en place des systèmes bureautiques manipulant l'information sous toutes ses formes (texte, voix, image), le recours aux spécialistes du traitement électronique de l'information (l'informatique, disent les francophones) s'impose à l'évidence.

La mise en place de systèmes bureautiques pourra être envisagée avec sérénité à deux conditions. Que, d'abord, les problèmes techniques soient pris au sérieux et résolus par les informaticiens, détenteurs actuels des compétences requises. Qu'ensuite, ces mêmes informaticiens n'oublient pas qu'il s'agit de créer des systèmes viables et réellement utiles aux usagers : près de 12 millions de cols blancs (en France), pour qui la bureautique naissante est d'abord perçue comme un bouleversement d'habitudes solidement ancrées. Sans l'adhésion de ces usagers, toute révolution bureautique serait vouée à l'échec.

Bernard Sauteur

Le présent numéro comporte un encart publicitaire folioté de la page 45 à la page 50.

A la recherche de la spécification idéale

La méthode de spécification idéale n'existe pas encore. Mais, dès aujourd'hui, des systèmes d'aide à la spécification sont disponibles : langages, documents de synthèse, analyseurs automatiques.

par Michel Demuynck et Bertrand Meyer

L'intérêt, pour l'informaticien, de disposer d'une authentique méthode de spécification a été clairement démontré dans un précédent article (la spécification : un outil pour l'informaticien ; « 01 Mensuel » n° 149, avril 1981, p. 62). L'étude qui suit tente de définir l'outil idéal et de porter à la connaissance du lecteur les réalisations d'ores et déjà opérationnelles.

Les critères d'une bonne méthode

Une méthode de spécification idéale, qui n'existe pas aujourd'hui, respecterait toutes les conditions suivantes :
— caractère statique : la méthode de spécification permet de décrire des problèmes plutôt que des solutions ;
— non-ambiguité : la notation utilisée se prête à une interprétation unique ;
— abstraction : les descriptions peuvent être effectuées à un niveau conceptuel suffisant ;
— généralité : toutes les classes d'applications peuvent être traitées ;
— modularité : la spécification de systèmes complexes peut être construite en plusieurs parties autonomes et réutiliser des éléments de spécifications antérieures ;
— syntaxe + sémantique : on peut décrire non seulement l'articulation d'un système en unités (sa « syntaxe »), mais aussi la fonction de chacune d'entre elles (leur « sémantique ») ;
— base théorique : le langage et la méthode de spécification reposent sur une base théorique définie rigoureusement ;
— clarté : le formalisme utilisé pour la spécification est clair et parlant ;
— facilité d'apprentissage : la méthode peut être maîtrisée sans effort démesuré ;
— facilité d'emploi : la méthode peut être mise en œuvre simplement ;
— système support : il existe un système automatique permettant la composition interactive des spécifications, leur archivage, leur validation, des vérifications diverses, et la production de documents (références croisées, index, etc.). Nous étudions plus loin la structure de ce type de « système d'aide à la spécification » (SAS) ;
— aide à la conception : la méthode produit des spécifications qui facilitent l'étape suivante du cycle de vie : la conception du système ;
— aide à la préparation des tests : la spécification fournit un guide pour la construction de jeux d'essais significatifs dans la mise au point des programmes qui seront obtenus ultérieurement.

De même que les qualités du logiciel, les qualités exigées des spécifications ne sont pas entièrement compatibles entre elles : le caractère statique s'oppose dans une certaine mesure à l'aptitude à guider la conception ; la clarté et la facilité d'apprentissage ne sont pas nécessairement compatibles avec la rigueur de la base théorique. Les critères précédents sont donc une référence par rapport à laquelle, en pratique, un compromis devra être trouvé.

Prendre des notes

Le premier mérite d'une méthode quelconque de spécification, et presque le principal, est d'exister. C'est en effet la première étape obligatoire de toute spécification qui est souvent la plus fructueuse : celle qui consiste à lire répétitivement le cahier des charges et à « prendre des notes » qui sont des réponses, exprimées dans un certain formalisme, aux questions que l'on pose au texte. Cette étape est une source d'enrichissement considérable dans la compréhension du problème.

Au fur et à mesure que l'on prend ces notes, on est tout naturellement amené à constituer un certain nombre de documents de synthèse. Nous verrons ci-dessous que certains « SAS » (systèmes d'aide à la spécification) permettent de les obtenir de façon en partie automatique. Ces documents, qui se révèleront fort précieux pour la suite du projet, sont en particulier :

— l'index : table des références à des notions techniques ;
— le glossaire : liste des définitions des termes correspondants ;
— la table des liens : donnant les références « croisées » entre les éléments de la spécification ;
— la trace : permettant de rapporter les éléments de la spécification aux éléments correspondants du cahier des charges.

Concrètement, ces quatre documents peuvent être partiellement regroupés. La rédaction de ce type de document, certes fastidieuse, surtout si elle est faite manuellement, apprend beaucoup sur le système étudié, sa cohérence, sa complexité, ses limites. C'est, semble-t-il, cette raison qui explique le succès de systèmes comme PSL/PSA, moins intéressants peut-être en tant que méthodes de spécification que comme systèmes d'aide à la documentation, soutenus par des analyseurs automatiques produisant des informations du type que nous venons de citer.

Un processus itératif

La rédaction du cahier des charges et celle de la spécification ne sont pas nécessairement des étapes successives et disjointes. Parfois, le cahier des charges n'existe pas vraiment, et doit être reconstitué à partir d'éléments assez vagues. Même dans le cas où il est présent, on s'aperçoit que tout effort de spécification systématique soulève de nombreuses questions qui ne trouvent pas de réponse dans le cahier des charges, et exigent en général des clarifications que seuls les « clients » peuvent apporter. Il s'agit souvent de points délicats mais importants qui avaient échappé aux rédacteurs initiaux, et qui les amèneront à repenser le problème. Il se produit donc en pratique une série d'aller et retour mutuellement enrichissants entre la spécification et le cahier des charges, et une série de contacts entre les responsables de ces deux documents. Si le formalisme de spécification est par trop éloigné du mode naturel d'expression des clients, le besoin peut, nous l'avons dit, se faire ressentir d'un « médiateur » capable de parler les deux langages.

Les systèmes d'aide à la spécification (SAS)

Lorsque les systèmes décrits sont de quelque importance, le texte de leur spécification, même modulaire et bien structuré, est un objet complexe ; de même qu'un texte de programme, il doit être géré de façon systématique si on veut pouvoir en assurer la mise à jour et l'évolution permanente tout en maintenant sa cohérence interne. Comme dans le cas des programmes, on désire aussi pouvoir réutiliser des éléments de spécifications précédemment réalisées, en constituant des bibliothèques de spécifications à l'image des bibliothèques de sous-programmes.

Une méthode de spécification quelconque doit donc s'accompagner, si elle est destinée à être utilisée de façon régulière et systématique, d'outils informatisés de gestion des spécifications. Plusieurs systèmes d'aide à la spécification (SAS en abrégé), existent ou sont en cours de réalisation. Ce sont des outils d'aide à la construction des systèmes informatiques qui se rapprochent par bien des côtés des systèmes de « programmation assistée par ordinateur » (PAO) dont un prototype est Mentor de

l'INRIA. Leur originalité vient de ce qu'ils se limitent en principe à des tâches de gestion et d'analyse de textes, à l'exclusion de tout aspect dynamique, c'est-à-dire dirigé vers l'exécution, telle la partie « production du code objet » des compilateurs. On notera cependant que certains SAS permettent de construire automatiquement des simulateurs des systèmes spécifiés ; cette possibilité est évidemment utile pour l'étude à priori du comportement d'un programme, mais risque de mettre en péril le caractère « statique » de la spécification.

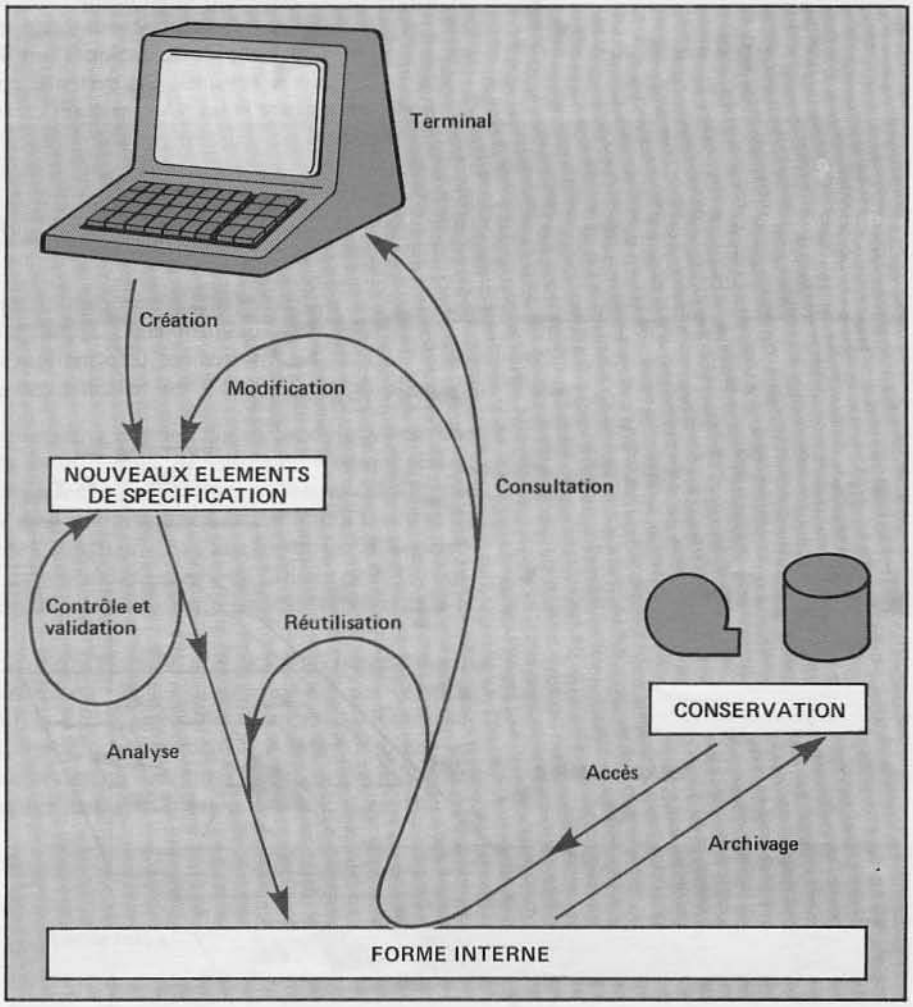


Figure 1 :
Structure générale d'un système d'aide à la spécification.

Bien que variant considérablement dans leurs détails, les différents SAS ont en commun une structure générale et un ensemble de fonctions schématisées sur la figure 1. Ces systèmes sont articulés autour d'une structure de données centrale, la forme interne, qui est une version des spécifications dépouillée des détails de syntaxe externe ; il peut s'agir, par exemple, d'« arbres abstraits » tels qu'ils sont utilisés par certains compilateurs comme forme interne des programmes.

Parmi les fonctions offertes par les SAS, on peut noter :

– la création de nouveaux éléments de spécification grâce à un éditeur de textes. Dans les SAS les plus évolués, cet éditeur connaît la structure du langage de spécification ;
– la modification de textes préparés antérieurement ;
– le contrôle et la validation des spécifications, permettant de détecter des erreurs et des incohérences (nom défini deux fois, élément de donnée produit par un composant du système mais non utilisé par un autre ou inversement, définitions contradictoires, objet non défini, etc.) ;
– l'analyse des spécifications pour leur traduction en forme interne ;
– la réutilisation d'éléments existants (cf. l'étape de reliure, ou « éditions de liens », en programmation) ;
– l'archivage d'éléments de spécification ;
– l'accès à des éléments archivés (opération inverse de la précédente) ;
– la consultation de la bibliothèque de spécification avec, par exemple, une recherche par mots clés.

A ces fonctions, s'ajoutent tous les éléments habituels liés aux outils de gestion de projets : procédures de mise à jour, de recherche, de contrôle d'accès, de documentation des modifications, etc. Seuls deux SAS ont, semble-t-il, atteint aujourd'hui le stade de l'utilisation industrielle : REVS dans le domaine du temps réel, PSA dans celui de la gestion. Il ne fait guère de doute que le succès de toutes les autres méthodes dépendra en partie de l'existence d'outils de qualité comparable.

A la découverte de l'existant

Nous terminerons cette étude de la spécification par une revue rapide de quelques-uns des systèmes de spécification les plus intéressants. Ces systèmes comprennent en général trois aspects : une méthode, un langage et un SAS, existant ou en projet. Dans certains systèmes, chacun de ces éléments possède un nom propre, et une certaine confusion en résulte dans l'emploi des sigles. Nous les désignerons sous la forme méthode (langage/SAS). Nous nous attacherons plus particulièrement à cinq systèmes : SA (SADT), SREM (RSL/REVS), ISDOS (PSL/PSA), HDM (SPECIAL) et Z.

SA (SADT) : Structured Analysis

SA se présente comme une méthode générale d'étude et d'analyse des problèmes, applicable à des situations très diverses, non nécessairement informatiques. Elle est en particulier utilisée comme méthode de spécification, en liaison avec un formalisme graphique appelé SADT

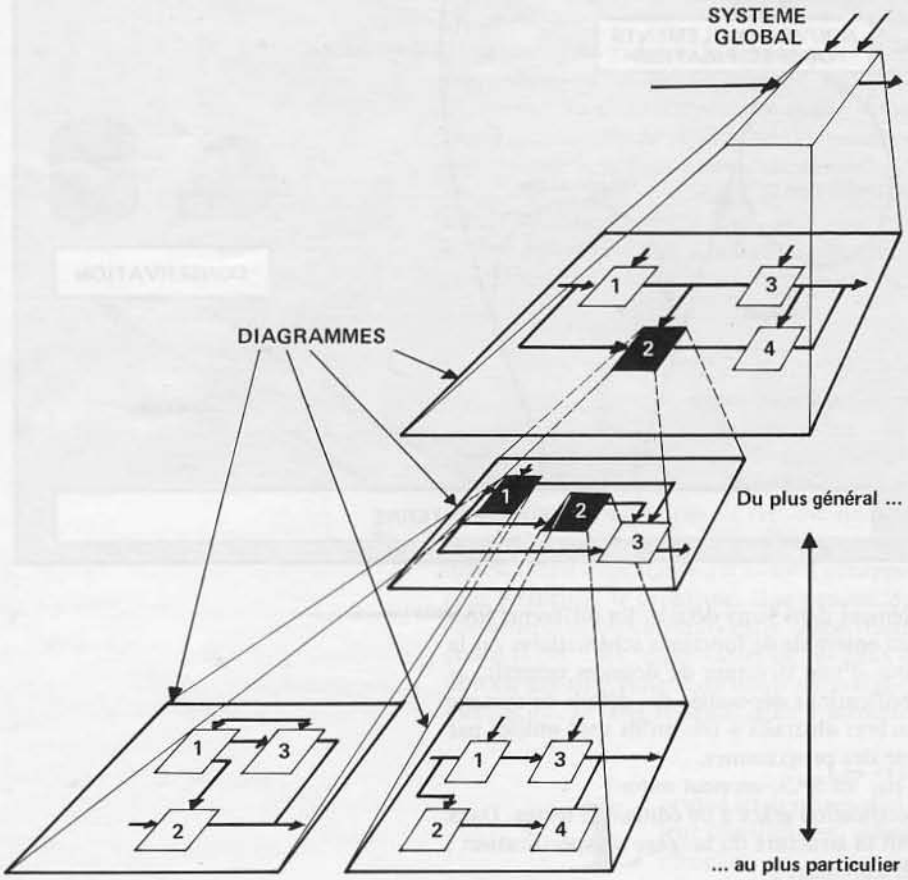


Figure 2 :
Structure d'une spécification en SADT.

(structured analysis and design technique). Des outils informatisés sont en préparation. SA est une méthode hiérarchisée et « descendante », qui repose sur le principe selon lequel toute « histoire » qu'on raconte, roman ou spécification, doit être présentée en trois chapitres :

— 1 : voici ce que je vais vous raconter ;
— 2 : le récit,
— 3 : voilà ce que je viens de vous raconter.

Dans ce schéma, le chapitre central, 2, a, en général, récursivement la même structure. On obtient donc une décomposition hiérarchisée des « récits » (figure 2). Chacun des éléments d'une telle décomposition est un schéma formé de

« boîtes » reliés par des flèches. Quatre types de flèches peuvent être associés à une boîte (figure 3) : « entrée », « contrainte » ou « mécanisme » (flèches entrantes), « sortie » (flèche sortante). Un schéma SADT peut comporter au plus six « boîtes » de ce type ; il décrit soit des « activités », soit des « données » ; ces deux concepts sont considérés comme duaux (figure 4).

SADT est commode pour montrer la décomposition hiérarchisée d'un système en blocs. On notera cependant que le formalisme manque d'une base mathématique rigoureuse et qu'il n'inclut pas la description de ce que nous avons appelé la « sémantique » du système, le rôle de chaque élément étant seulement indiqué par des noms ou des phrases en anglais.

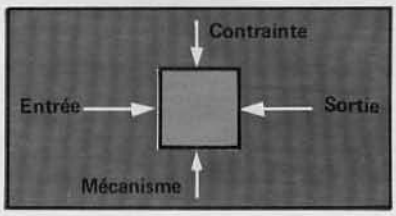


Figure 3 :
Types de flèches en SADT.

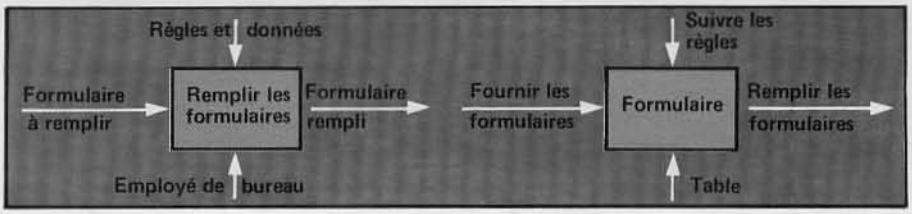


Figure 4 :
Dualité activités/données en SADT.

SREM (RSL/REVS)

SREM (software requirements engineering methodology) est une méthode d'aide à la spécification de systèmes « temps réel », développée par TRW pour des applications militaires (guidage de missiles). SREM est fondé sur l'emploi de diagrammes appelés R Nets, qui tiennent de l'organigramme de fonction et du réseau de Petri (figure 5) ; une forme linéaire correspondante, le langage RSL (requirements statement language), est utilisée pour la communication avec le SAS associé. La figure 6 montre le texte de RSL représentant le R Net de la figure 5.

SREM s'accompagne d'un SAS très élaboré, appelé REVS (requirements engineering and validation system), utilisé en particulier pour effectuer des vérifications systématiques des spécifications. RSL et REVS se veulent extensibles : les utilisateurs peuvent, sous certaines contraintes, ajouter des primitives. REVS comprend un générateur de simulateurs permettant de tester les systèmes sous forme abstraite. SREM paraît avoir subi avec succès l'épreuve de l'utilisation industrielle. Les réserves

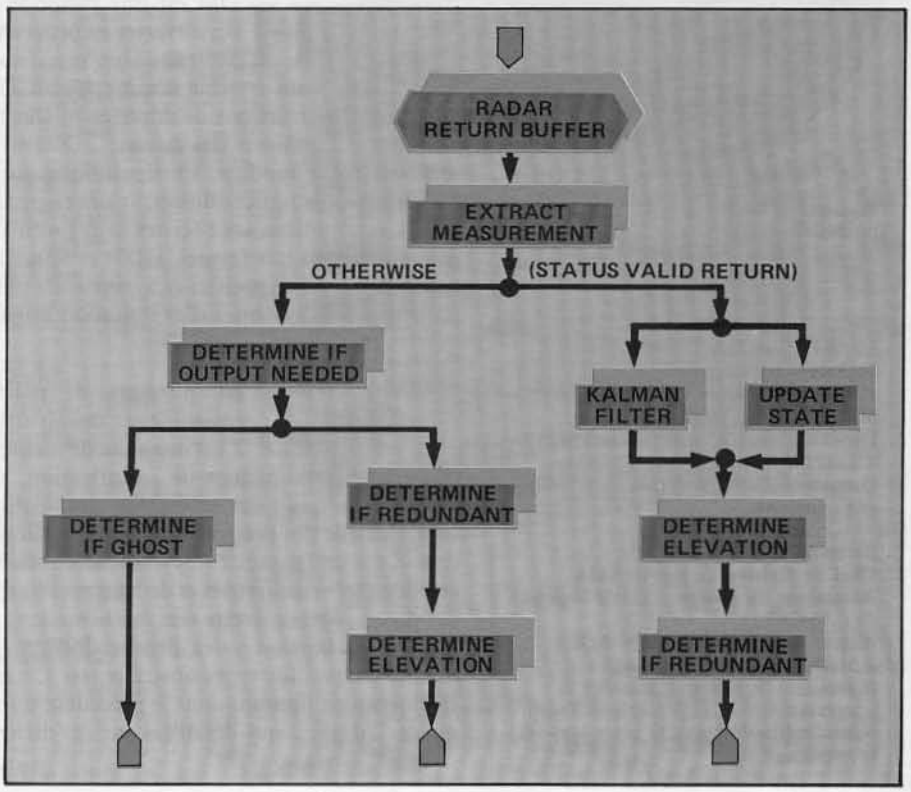


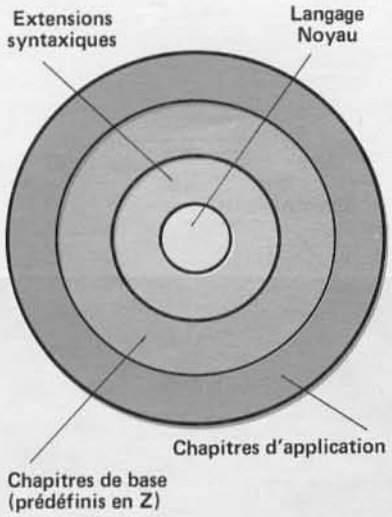
Figure 5 :
Un diagramme (R Net) de SREM.

qu'on peut émettre concernent le caractère assez procédural et dynamique de cette méthode très orientée vers la notion de processus et d'action, et qui, par certains aspects, se rattache à la conception autant qu'à la spéficiation.

```
R NET      PROCESS RADAR RETURN
     STRUCTURE
          INPUT INTERFACE RADAR RETURN BUFFER
          EXTRACT MEASUREMENT
          DO (STATUS VALID RETURN)
               DO  UPDATE STATE AND KALMAN FILTER  END
               DETERMINE ELEVATION
               DETERMINE IF REDUNDANT
               TERMINATE
          OTHERWISE
               DETERMINE IF OUTPUT NEEDED
               DO   DETERMINE IF REDUNDANT
                    DETERMINE ELEVATION
               AND  DETERMINE IF GHOST
                    TERMINATE
               END
          END
     END
```

Figure 6 :
Un élément de spécification en RSL.

Figure 7 :
Structure du langage Z.



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.

ISDOS (PSL/PSA)

ISDOS, développé depuis le début des années soixante-dix à l'université du Michigan, s'adresse plutôt au domaine de l'informatique de gestion. Le but poursuivi est la documentation des grands projets de gestion, qui sont décrits dans le langage PSL (problem statement language) à l'aide d'entités et de relations (échange de données, de messages) entre ces entités. Le SAS associé, appelé PSA (problem statement analyzer), effectue un grand nombre de vérifications. ISDOS et le langage PSL ne couvrent qu'une partie relativement limitée de la spécification : la documentation des relations entre composants d'un système. Dans ces limites, cependant, la méthode fournit un ensemble de règles et d'outils qui ont su trouver des utilisateurs.

HDM (SPECIAL)

HDM est une méthode fondée sur l'une des principales notions découvertes en programmation ces dernières années, celle de type abstrait. Une spécification en HDM, exprimée dans le langage Special, décrit un système comme formé d'un certain nombre de modules construits autour d'« abstractions de données ». Des fonctions de divers types (V-fonctions qui ne font que consulter des données, O-fonctions qui en créent) sont définies relativement à chaque module. La sémantique du système est exprimée par des assertions : précondition et postcondition, vraies respectivement avant et après l'appel d'une fonction. Ces assertions sont écrites sous forme de prédicats logiques, ce qui donne au langage son aspect très formel. HDM et Special ont été relativement peu diffusés ; ils ont cependant servi à spécifier et à construire plusieurs systèmes de type « temps réel », en particulier des systèmes d'exploitation à hautes exigences de fiabilité.

Le langage Z

Z est un langage de spécification formel reposant sur la théorie des ensembles et des structures mathématiques. Une spécification en Z est une suite de « chapitres », destinés à être archivés pour constituer des bibliothèques de spécification. Chaque chapitre contient des définitions d'objets tels que : des ensembles, des relations, des fonctions, des théorèmes (expression formelle des propriétés du système modélisé) ou des classes (ou structures abstraites). Z est un langage extensible par couches successives (figure 7). Z permet des descriptions très rigoureuses et de haut niveau d'abstraction. Il a été appliqué à des domaines divers. Son caractère très mathématique en fait cependant un outil difficile à généraliser. Un langage assez proche de Z par son aspect formel, fondé sur la théorie mathématique des types abstraits, est Clear. Le lecteur voudra bien se reporter à la bibliographie figurant dans le précédent numéro de « 01 Mensuel » (p. 66) pour obtenir des informations détaillées sur ces différentes méthodes.

Michel Demuynck et Bertrand Meyer