Construction raisonnée des programmes

Plusieurs approches ont été suivies pour rendre plus rigoureux le processus de construction des programmes informatiques. Si ces avancées ont permis d'améliorer la pratique industrielle, le problème de la construction de programmes corrects reste encore très largement un défi actuel.

La programmation des ordinateurs était à l'origine un processus artisanal, avec une démarche d'essai-erreur-correction. Avec l'augmentation de la taille et de la complexité des programmes, cette façon de faire a rapidement trouvé ses limites.

La décomposition d'un problème en sous-problèmes plus simples est à la base de nombreuses méthodes de développement de programmes. Dès le début de la programmation (vers 1950-51) a été introduite la notion de sous-programme. Si elle permet de regrouper dans une unité séparée un ensemble d'opérations fréquemment réutilisées, elle ne résout pas le problème de la structuration d'ensemble et n'impose pas de discipline de programmation.

Vers 1967-68 apparut la notion de décomposition en couches, appelées «machines abstraites». Chaque couche résout un problème spécifique en s'appuyant sur les couches inférieures, la couche la plus basse étant l'ordinateur lui-même. Cette méthode fut notamment développée par Edsger Dijkstra et appliquée avec succès à la construction d'un système d'exploitation, THE.

Dans les années 1969-1972, plusieurs avancées furent réalisées :

- Robert Floyd, puis C. A. R. Hoare introduisirent des éléments de sémantique des programmes, en spécifiant l'effet d'une séquence de programme sous forme de pré- et post-conditions (partant de tel état «avant», et si telle condition est remplie, on arrive à tel état «après»). Cette méthode fournit une base à la fois pour la spécification et pour la preuve de programme.

- D'autres chercheurs (notamment David Parnas) s'intéressèrent à la décomposition d'un programme complexe en «modules». La difficulté ici est de trouver une décomposition qui aide au travail de conception et qui facilite l'évolution ultérieure du programme. Plus tard, la notion de module fut introduite dans des langages de programmation.

- Les méthodes de construction par raffinements successifs de «machines abstraites» furent améliorées grâce à l'introduction d'éléments de sémantique, notamment par Barbara Liskov et par Jean-Raymond Abrial (méthode Z, dont devait plus tard dériver la méthode B).

Dans les années 1990 se répandit l'usage de «patrons» (en anglais *patterns*), schémas d'architecture logicielle répondant à une situation spécifique, dont de nombreux catalogues ont été publiés. Un «canevas» (en anglais *framework*) est la mise en œuvre d'un patron au moyen d'un programme dans un langage particulier.

Toutes ces idées, enrichies au cours du temps, sont à la base des méthodes actuelles, mais elles ne résolvent pas tous les problèmes et leur pénétration dans la pratique industrielle reste lente. Aujourd'hui, beaucoup de programmes sont encore écrits de manière empirique, même si l'usage de méthodes raisonnées progresse, par exemple par la relecture systématique du code et par la large diffusion d'outils de conception tels qu'UML, qui facilite notamment l'usage de patrons..

Sur le front de la recherche, l'usage d'assistants de preuve permettant le développement conjoint d'un programme et de sa preuve de validité conduit à des résultats très encourageants, mais sa diffusion dans la pratique industrielle semble encore très lointaine.