Université Pierre & Marie Curie Maîtrise d'informatique

Programmation
Spécification et certification du logiciel

Notes de cours et exercices

P. MANOURY B. PAGANO

1998-99

Résumé

Ce cours est une introduction à l'utilisation des *méthodes formelles* dans le développement du logiciel. Les méthodes, dites *formelles*, de spécification et validation du logiciels reposent sur l'utilisation dans le domaine de l'informatique des travaux développés depuis le début de ce siècle dans le domaine de la logique mathématique. Elles en héritent rigueur et précision qui sont autant d'avantages pour le contrôle de la *qualité* et de la *correction* du logiciel.

Le cours comprend en trois parties :

1. un rappel indispensable des bases de la *logique formelle* ainsi que de la *théorie des ensembles.*
2. une présentation de la *méthode des assertions* initiée par Hoare et Dijkstra qui permet une certification *a posteriori* des programmes impératifs.
3. une présentation de la *méthode B* due à Abrial qui met en œuvre la technique du développement de programmes par *raffinement* et qui permet une certification *a priori* des programmes engendrés.

L'objectif du cours est de donner aux étudiants une maîtrise

1. de la formalisation des problèmes destinés à être tarités par des systèmes logiciels,
2. des méthodes de certification formelle que seule autorise à la fois la formalisation préalable des problèmes et l'intelligence théorique des mécanisme de base des programmes.

Méthodes formelles

On a coutume de diviser le *cycle de vie* d'un logiciel en deux grandes phases : une phase d'élaboration suivie d'une phase de validation. Chacune de ces phases est elle-même subdivisée en plusieurs étapes.

1. La phase d'élaboration comprend :
   (a) la rédaction d'un cahier des charges exprimant les besoins,
   (b) la rédaction d'une spécification générale précisant le *"quoi faire"* pour satisfaire ce besoin,
   (c) l'élaboration d'une spécification détaillée indiquant le *"comment faire"*,
   (d) le codage des composantes du logiciel.
2. Une fois le code engendré, on procède aux trois étapes suivantes :
   (a) soumission du code à un certain nombre de tests unitiares ;
   (b) intégration du logiciel dans son environnement d'exploitation (réel ou simulé),
   (c) recette finale.

L'utilisation des méthodes formelles tout au long de ce cycle de vie a d'importantes incidences tant sur la *qualité* du processus de développement que sur la *correction* du code obtenu.

Les méthodes formelles empruntent leurs outils à la *logique formelle* développée essentiellement dans la première moitié de ce siècle. Elles en héritent la rigueur et la précision qui sont autant d'avantages pour le contrôle de la qualité et de la correction que nous évoquons ci-dessus.
Rigueur et précision sont dues à deux constituants majeurs de la logique formelle :

un *langage* qui élimine l'ambiguïté toujours possible des langues naturelles ;

un formalisme de *preuves* permettant de garantir la validité des énoncés logiques.

L'avantage essentiel de la langue formelle est qu'elle est un référentiel pour l'ensemble des équipes intervenant dans le développement. Plus précisément, on peut relever plusieurs points d'impact (bénéfique) de l'emploi des méthodes formelles sur le cycle de vie du logiciel.

1. Le langage de la logique peut-être utilisé lors de la seconde phase du processus d'élaboration : reformulation du cahier des charges en une spécification générale. L'expérience montre que cet effort de formalisation permet de lever l'ambiguïté d'interprétation, l'imprécision et bon nombre d'omissions que contient inévitablement l'expression d'un besoin en langue naturelle. On élimine ainsi une grande partie des sources d'incohérences et d'erreurs dans le codage.
2. Si la spécification générale et la spécification détaillée sont écrites formellement, la cohérence du passage de l'une à l'autre peut être assurée avec exactitude. Il s'agit dans ce cas de valider le passage d'un énoncé formel à un autre. Ce qui se réduit à la démonstration qu'un énoncé logique se déduit d'un autre. On peut mener à bien, et avec sûreté, cette tâche par l'emploi de preuves formelles.
3. De façon analogue, le passage de la spécification détaillée au programme l'implémentant met en jeu deux formalismes : le langage logique et le langage de programmation. Dés lors que le langage de programmation est bien compris (*ie* : modélisé), la preuve formelle permet d'assurer la validité du passage d'un formalisme à l'autre.
4. Enfin, puisque l'on a l'assurance que le programme obtenu répond bien à une spécification précise dont ont été éliminé toutes sources d'ambiguïté ou d'omission, le travail de test s'en trouve considérablement allégé : le composant logiciel étant *en lui-même* validé, il ne s'agit plus que de procéder aux tests d'intégration.

En ce qui concerne la *qualité* de la construction et la *correction* du logiciel, l'emploi des méthodes formelles, et en particulier l'emploi de preuves formelles, apporte donc des critères de garantie absolus.

Qualité : la vérification de la correction d'une preuve formelle est un processus qui s'apparente à l'analyse syntaxique, ce qui permet donc de décider si oui ou non une preuve est une preuve, indépendemment de son contenu.

Correction : le calcul effectué par le code est une conséquence logique de la spécification formelle initiale, la légitimité du passage de celle-ci à celui-là est donc garantie par la validité même des preuves formelles.

Si elles offrent un avantage indéniable, les méthodes formelles ont cependant un certain coût. La formalisation exhaustive des spécifications, la conduite rigoureuse des preuves de validation est un travail non trivial, long et souvent fastidieux. Dans la pluspart des cas, l'investissement à consentir pour l'utilisation de ces méthodes semble exagéré en regard des avantages qu'elles procurent. D'autant que l'industrie du logiciel posséde et utilise en effet nombre d'autres méthodes visant à assurer tant la qualité que la correction des systèmes informatiques. Cependant, ces méthodes sont, pour la plupart, héritées d'autres processus de fabrication et, par tant, ne prennent pas en compte la spécificité de la production du logiciel qui est, avant tout, une activité intellectuelle : un programme est d'abord un *texte*.

L'industrie, et on peut le comprendre, m'aime pas lacher la proie pour l'ombre et c'est pourquoi, à ce jour, seuls certains secteurs industriels pour lesquels l'exigence de qualité de développement et l'impératif de sûreté de fonctionnement sont de prime importance ont intégré les méthodes formelles dans leur processus de développement. Citons, pour l'essentiel, les industries du transport, ferroviaire ou aérien, les industries de l'énergie ainsi que le secteur bancaire.
L'expérience menée dans ces domaines s'est concentrée sur la certification des parties les plus sensibles à la sécurité des système logiciels : systèmes de freinage d'urgence ou de pilotage automatique, systèmes de contrôle de commandes asservies, protocoles d'autentification de transactions, etc. Non seulement,

l'emploi des méthodes formelles s'est avéré fructueux pour les problème traités, mais encore, l'expérience acquise permet d'envisager de ne plus restreindre cet emploi au seul développement des parties les plus sensibles des systèmes logiciels.
Enfin, le développement du réseau fait apparaître de nouveau besoin en certification de code. Nombre détudes sont menées à l'heure actuelle pour intégrer au code lui-même la preuve de certaines de ses propriétés (consommation de ressource limitée, droit d'appels systèmes limités, etc ..) Ce code contenant sa certification prend son sens dans les applications mettant en œuvre la distribution et la migration de code : il garantit qu'un code herbergé par une machine distante ne causera pas de dommage ou ne portera pas atteinte à son intégrité. Ici encore, les méthodes formelles sont les candidates naturelles à la mise en œuvre de ces techniques car la machine ne peut comprendre que du langage entièrement formalisé.
Ces résultats laissent augurer le développement à moyen terme de l'emploi des méthodes formelles dans la production de logiciel.

RÉFÉRENCE : sur le sujet des méthodes formelles en général, le lecteur pourra lire avec profit le rapport et les recommandations que l'Observatoire Français des Techniques Avancées (OFTA) a publié en 1997 sous le titre : *Application des techniques formelles au logiciel.* Ce volume contient, en particulier et en annexe, une courte présentation de la méthode B.