milliers de voyageurs quotidiennement peut susciter de légitimes appréhensions. « *La RATP trouvait que les méthodes de sécurité traditionnelles étaient insuffisantes* », confie Luis-Fernando Meija, responsable des méthodes formelles chez Alstom. Le choix s'est porté sur la méthode B pour garantir au transporteur la fiabilité des logiciels « sécuritaires » (liés à la sécurité des passagers) du Sacem. « *Il a été motivé par les difficultés inhérentes à la validation du Sacem* », confirme Pierre Desforges, responsable de l'équipe sûreté de fonctionnement de l'unité ingénierie du transport ferroviaire du département des équipements et systèmes du transport de la RATP.

▶ Le recours aux outils est un passage obligé

« *Les applications embarquées sont un défi en matière de sécurité,* témoigne, pour sa part, Daniel Le Métayer, directeur technique de Trusted Logic, une société spécialisée dans le logiciel sur cartes à puce. *Nous utilisons la méthode formelle Coq pour vérifier le code chargé sur les puces.* » A mesure que les exigences des utilisateurs vis-à-vis des logiciels croissent, les méthodes formelles deviennent donc peu à peu incontournables.

Leur emploi n'est cependant pas simple. « *Il n'est pas possible d'utiliser B sans outils* », explique Luis-Fernando Meija. Et, au début des années quatre-vingt-dix, il n'en existait aucun. Candidat à la réalisation du projet Météor (quatorzième ligne du métro parisien, entièrement automatisée), Alstom a constitué une équipe pour l'industrialiser. Mais c'est Matra Transport International (MTI) qui a finalement remporté l'appel d'offres. Néanmoins, durant deux années, un ensemble d'outils a tout de même vu le jour, constituant peu à peu l'Atelier B, aujourd'hui maintenu et commercialisé par Clearsy. Cet atelier de modélisation et de preuve permet d'automatiser certaines tâches durant le cycle de développement. Il assure, entre autres, la vérification syntaxique des modèles, la génération automatique de théorèmes, la preuve de ceux-ci, ou encore la traduction automatique des modèles B en langages C ou Ada. La démarche a été comparable chez Trusted Logic : « *Coq est un composant qui complète le cadre offert par notre outil TL-FIT* », explique Daniel Le Métayer. Ce produit intègre, lui aussi, les différents niveaux de raffinement et les vérifications de cohérence exigées par les « critères communs de sécurité », norme internationale utilisée pour la certification des logiciels (voir, à ce sujet, notre enquête parue dans le numéro 1657 du 30 novembre 2001, page 30).

▶ Un surcoût compensé par une utilisation élargie

Mais, pour attirer de nouveaux adeptes, les seuls arguments techniques ne suffisent pas. Comme partout, au final, c'est le facteur économique qui prime. « *Les méthodes formelles entraînent un surcoût en phase de développement, atténué par les gains réalisés en phase de validation. Et ce pour des niveaux de qualité et de sécurité estimés bien meilleurs* », argumente Pierre Desforges. Et pour Jean-Marc Meynadier, responsable du service équipement pilote automatique chez MTI, « *la réutilisation permet d'en réduire fortement les coûts* ». Ces arguments pourraient bientôt faire sortir les méthodes formelles du seul domaine des logiciels sécuritaires embarqués. « *Nous évaluons actuellement la possibilité d'utiliser B pour le développement des logiciels non sécuritaires* », confirme Jean-Marc Meynadier. De son côté, Thierry Servat plaide pour l'extension de son champ d'application aux systèmes dans leur globalité. « *L'intérêt, c'est de monter le plus haut possible dans la spécification,* affirme-il. *Il faut considérer le système comme un tout, et ne pas essayer de séparer le logiciel du matériel. C'est de cette manière que nous pouvons diminuer substantiellement les coûts.* » Une démarche qui, si elle devait être mise en œuvre, permettrait, de plus, de limiter les tests d'intégration tout en garantissant une plus grande maintenabilité des systèmes. Car les méthodes formelles permettent naturellement d'améliorer la traçabilité des développements. La RATP envisage d'ailleurs d'assurer désormais elle-même la maintenance de ses logiciels. « *On maîtrise mieux l'impact des modifications apportées aux spécifications* », assure Pierre Desforges.

Jean-Marie Portal

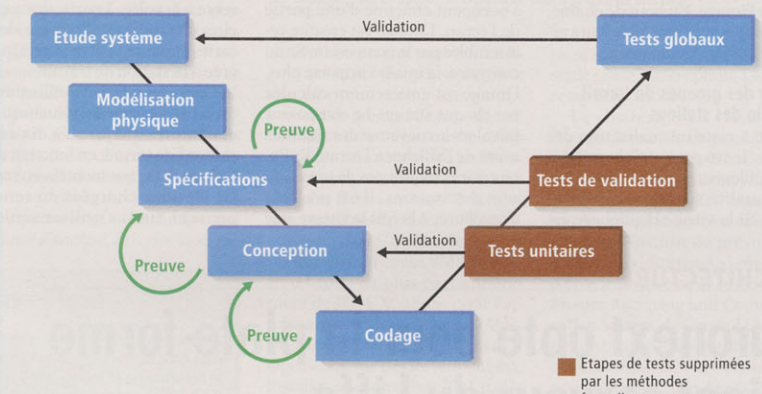
Technologie

Les méthodes formelles ont un impact sur le cycle de développement logiciel

La suppression des tests unitaires et de validation est conditionnée à l'emploi de la preuve tout au long du cycle.

Contrairement aux tests, qui n'interviennent qu'après la phase de codage, les méthodes formelles avec preuves accompagnent l'ensemble du cycle de développement logiciel. Y compris lors du passage du cahier des charges – ou modélisation physique – à la spécification technique, souvent considéré par les experts comme le maillon faible de la chaîne. Car la transcription de l'un vers l'autre dans un langage différent augmente le risque d'erreurs. De fait, il est généralement nécessaire de reformuler le cahier des charges, en vue de sa formalisation, au sein d'une spécification générale. Celle-ci est ensuite décomposée en propriétés successives pour constituer au final une spécification technique structurée. Durant le déroulement de cette étape, dite de raffinement, il est prouvé que chaque propriété – ou modèle – nouvellement construite n'introduit aucune incohérence avec celle de niveau supérieur. Le résultat est la formalisation de toutes les propriétés du système. L'étape suivante consiste alors à transformer les modèles abstraits obtenus en modèles concrets. Ceux-ci contiennent des variables pouvant être traduites dans un langage informatique classique. La preuve, qui intervient tout au long de ce processus, est l'élément essentiel qui remet en cause les tests. Elle garantit non seulement la cohérence des éléments constitutifs du système, mais aussi que, lors de l'introduction de chacun d'entre eux, aucun ne remet en cause le travail déjà effectué. Les adeptes des méthodes formelles estiment que la preuve offre une plus grande garantie d'exhaustivité que celle procurée par les tests, qualifiés d'approximatifs. Dans un cycle en V, la preuve a un impact sur les parties basses du cycle. Elle autorise donc la suppression des tests associés aux phases de spécification et de conception – soit, respectivement, les tests de validation et les tests unitaires. Mais les tests globaux demeurent toujours d'actualité. Ils permettent de valider, pour leur part, la partie haute du cycle en V. En l'occurrence, le cahier des charges.

Le cycle en V corrigé par la preuve



POUR EN SAVOIR PLUS

The B-Book : Assigning Programs to Meanings, par Jean-Raymond Abrial ; Cambridge University Press ; 1996
L'ouvrage de référence de la méthode B.

www-lsr.imag.fr/B
Informations sur la méthode B, ses outils et ses utilisateurs.

www.fmeurope.org/
Site de l'organisation Formal Methods Europe.

QUESTIONS/RÉPONSES

▶ L'utilisation des méthodes formelles nécessite-t-elle des compétences particulières ?
Parce qu'elles font généralement appel à des concepts mathématiques, les méthodes formelles nécessitent une double compétence en mathématiques et informatique. Néanmoins, certaines entreprises, telle la RATP dans le cadre du projet Météor, n'hésitent pas à dispenser des formations internes à leurs informaticiens pour les initier à cette technologie. Et cela semble marcher.

▶ Toutes les méthodes formelles sont-elles équivalentes ?
Les méthodes formelles avec preuve se distinguent des autres en ce qu'elles sont les seules à permettre aujourd'hui de supprimer un certain nombre de tests, dont les tests unitaires. Mais c'est ensuite l'outillage disponible pour les mettre en œuvre qui permet de les différencier. Par exemple, la méthode B a ceci de distinctif qu'elle est aujourd'hui l'une des seules à pouvoir couvrir l'ensemble du cycle de développement logiciel, jusqu'à la production du code. Et cela grâce à l'Atelier B.

AVIS D'EXPERT


G. FONLUPT

Jean-Raymond Abrial, consultant indépendant, inventeur de la méthode B.

« Il y a lieu d'être prudent »

L'emploi de la preuve garantit qu'il n'y aura pas de distorsion entre le modèle mathématique initial, issu de la spécification écrite en français, et le modèle mathématique final, extrêmement proche du code exécutable. Mais, en deçà et au-delà, rien n'est garanti par la preuve. Si, par exemple, la spécification en français est erronée ou si le modèle mathématique initial n'est pas fidèle à cette spécification, on sort des limites de l'épure. De même, si le traducteur qui transforme le dernier modèle mathématique en code ou si le compilateur qui le traduit en code binaire sont bogués, on ne peut rien garantir. Le zéro défaut absolu n'existe pas. Mais on peut s'en approcher toujours plus. Ce que l'on peut faire avec une méthode formelle va certainement beaucoup plus loin que ce que l'on peut garantir avec des tests unitaires. Mais il ne faut pas croire que le problème est résolu dans son ensemble : le retour de bâton serait terrible.