Logiciels : les objets à l'épreuve des faits

Marie-Claude Gaudel, Professeur,
Université Paris Sud 11,
LRI (Laboratoire de Recherche en Informatique),
Bâtiment 490,
91405 Orsay-cedex
mcg[at]lri[point]fr

Les applications informatiques, ainsi que les machines et systèmes qui les supportent sont de plus en plus complexes. Dans ce contexte, il devient de très difficile de développer et de garantir les logiciels. La réalisation de logiciels de qualité est à la fois un objectif économique essentiel et un axe de recherche de longue haleine –le Génie Logiciel– qui combine des activités très théoriques et très appliquées.
Les enjeux économiques de ce domaine sont tels qu'ils en perturbent parfois les avancées scientifiques.

1 - État des avancées scientifiques dans les dix dernières années

Depuis un peu plus de dix ans les « méthodes orientées objets [1]» se sont taillées une place dominante dans le monde du Génie Logiciel, dans le discours des consultants et des scientifiques du domaine.
La définition de standards, sur des bases empiriques, par l'OMG[2], a eu pour effet d'entraîner toute une communauté scientifique vers une recherche sur la compréhension de ces standards, et des méthodes orientées objets qui en étaient le credo de départ.
Même si il existait plusieurs autres tentatives, bien fondées scientifiquement, de méthodes de développement de logiciel, aucune n'était vraiment passée dans la pratique. Il semblait donc constructif de les confronter et de les faire évoluer en fonction des spécifications produites par l'OMG, en particulier UML qui est un ensemble de notations permettant de décrire des logiciels en cours de développement.
Avec le recul, on se trouve cependant confronté à un paradoxe. Programmation, Analyse et Conception Orientées Objets ont apporté autant de problèmes scientifiques qu'elles en ont résolus. L'OMG continue à produire des spécifications de plus en plus complexes (MDA, UML Profiles, MOF, …). Le monde de la recherche continue à chercher à leur donner du sens et à développer les théories et les techniques pour permettre de garantir la qualité des logiciels produits.

---

[1] Par « méthodes orientées objets », on entend ici tout ce qui concerne l'analyse (définition de ce que le logiciel doit faire), la conception (décisions de comment il le fera) et enfin la programmation. Les méthodes de validation et vérification, maintenance, réutilisation sont aussi considérées, même si elles sont moins développées, dans le cadre orienté objets, que les précédentes.

[2] L'OMG, Object Management Group, a été fondé en 1989 par un consortium de 11 industriels vendeurs de méthodes et d'outils de Génie Logiciel.L'objectif en était : *« the establishment of industry guidelines and detailed object management specifications to provide a common framework for application development. ».* Ce consortium regroupe actuellement plus de 800 industriels du Génie Logiciel et émet régulièrement des spécifications de méthodes, langages et outils. Voir www.omg.org.

1.1 - Modularité et réutilisation

Si on consulte la littérature sur les méthodes orientées objets à leur début, les motivations que l'on retrouve le plus souvent sont la modularité (développement et validation indépendants de parties de logiciels) et la réutilisation.
En fait, la décomposition en objets ou en classes d'objets s'est avérée correspondre à une granularité très petite et n'a pas apporté d'amélioration vraiment significative de ces points de vue. Cela explique d'une part la présence dans UML de la notion de « paquetage », qui correspond à des modules logiciels ou à des sous-systèmes, ainsi que le développement de recherches dans deux directions :

- Des modèles et des langages de description d'architecture logicielle, qui, au niveau de la conception, fournissent des archétypes et des outils de descriptions de composants logiciels, de connecteurs et d'organisations architecturales globales.

- Des patrons de conception (design patterns) qui masquent les difficultés de la programmation à objets en fournissant des modèles efficaces pour les constructions fréquentes dans ce type de programmation.

1.2 – Distribution et objets

Les problèmes de réalisation des applications distribuées à base d'objets ont été également à l'origine du développement des ORB (Object Request Broker), qui permettent de gérer les noms et les accès à des objets distants. En effet, ces objets font abondamment référence les uns aux autres et toute reconfiguration, toute mobilité, devient vite techniquement très complexe.
Ces ORB, dont CORBA est le modèle le plus connu, sont intégrés dans ce qu'on appelle du « middleware » ou de l'intergiciel : il s'agit de couches logicielles, intermédiaires entre les systèmes d'exploitation et les applications, qui permettent une gestion transparente de la distribution en général, et des objets distribués en particulier.

2 - Prospective à court terme sur quatre ans

On est loin d'avoir résolu toutes les questions de recherche soulevées par les problèmes ci-dessus.
Architectures, patrons, composants et composition, restent donc des sujets d'actualité. Il faut y ajouter les méthodes de validation, de vérification, et d'évaluation des systèmes à objets qui ne sont que très peu développées à cause du manque de définitions sémantiques précises des notations et des méthodes proposées par l'OMG.
Il faut mentionner l'importance prise par les intergiciels sur lesquels repose en grande partie la sûreté de fonctionnement des systèmes ainsi développés et la possibilité de les faire coopérer.

3 - Inventaire des problématiques les plus actuelles

Les problématiques les plus actuelles sont souvent anciennes, reprises dans le contexte de l'orienté objet. On peut citer, entre autres :

- Quelles compositions pour quels composants logiciels?

• Comment faire de la vérification compositionnelle, c'est-à-dire de la vérification composant par composant qui assure des propriétés globales d'un ensemble logiciel ?

D'une manière générale, même si plusieurs tentatives intéressantes ont eu lieu, on manque encore cruellement de sémantique formelle des modèles et des programmes orientés objets, donc d'outils de transformation, de validation et de vérification bien fondés.
L'expérience de ces dernières années a clairement montré que pour avancer sur ces derniers points il faut, certes, encourager les recherches dans ce domaine, mais qu'il est essentiel de continuer à progresser par ailleurs sur les approches formelles du Génie Logiciel et sur les modèles de systèmes distribués.
Par exemple, certains outils de preuve de programme en Java (donc orienté objet) commencent par construire un programme fonctionnel équivalent : cela permet d'exploiter le riche corpus de résultats scientifiques en sémantique de la programmation fonctionnelle.
D'autres approches ramènent les documents d'analyse et de conception d'UML à des descriptions basées sur des modèles à base d'automates, ou des types abstraits de données, afin d'utiliser des résultats de recherche qui ont été pendant un temps considérés comme démodés car trop éloignés des recommandations de l'OMG.

4 - Forces et les faiblesses de la communauté scientifique française par rapport aux autres communautés

En ce qui concerne les approches orientées objets, il existe une communauté de recherche active, mais relativement dispersée et peu organisée, pas toujours unanime dans ses certitudes scientifiques. Certains membres de cette communauté sont actifs au sein des organismes de standardisation et assurent ainsi une évolution positive des méthodes et outils de Génie Logiciel vers plus de rigueur scientifique.

Les communautés de recherche sur les approches formelles et les systèmes distribués sont très actives en France et reconnues comme de tout premier plan au niveau européen et international. Certaines équipes ont établi des contacts industriels fructueux et sont à l'origine de beaux succès techniques. N'oublions pas que les Airbus, le TGV, et la plupart des réalisations techniques avancées contiennent beaucoup de logiciels dits « embarqués ».
La communication entre ces communautés existe et fonctionne, mais vu les enjeux, il faut l'encourager.

Références Générales

Formal Methods Europe, *Virtual Library on Formal Methods*, http://www.fmeurope.org/

Richard P. Gabriel , *Objects have failed, Notes for a debate*, ACM Conference on Object Oriented Programming, Systems and Applications (OOPSLA), nov. 2002, www.dreamsongs.com/NewFiles/ObjectsHaveFailed.pdf

Marie-Claude Gaudel , Bruno Marre, Françoise Schlienger, Gilles Bernot. *Précis de Génie Logiciel*, Masson, 1996

J. Rumbaugh et al.,*The Unified Modeling Language, the Reference Manual*, Prentice Hall, 1998

Ph. Schnoebelen, ed.. *Vérification de logiciels : Techniques et outils du model-checking*. Vuibert. 1999

Sigles

OMG : Object Management Group
UML : Unified Modeling Language
MDA : Model Driven Architecture
MOF : Meta Object Facility
ORB : Object Request Broker