Journées doctorales ENSTA
27 Janvier 2009

A quoi correspond le travail d'un doctorant en vérification de système ?

Tout d'abord, oubliez tous les clichés... Nous ne passons pas 15 heures par jour derrière un clavier à écrire du code dans des langages incompréhensibles. Comme dans de nombreux domaines, nous passons plusieurs heures par semaine à effectuer des recherches bibliographiques ou à assister à des présentations organisées dans un lieu proche de notre laboratoire, pour savoir ce qui se fait dans les autres équipes du monde, et pour vérifier si la nouvelle idée « géniale » que nous venons d'avoir n'aurait pas déjà discrètement échoué il y a quelques années. La majeure partie du temps restant est consacré à la conception et à l'analyse d'un système, ce qui demande surtout un gros travail mathématique.

Quelles sont les difficultés auxquelles sont confrontées les doctorants ?

A priori, le travail d'un doctorant est différent de tout ce que vous avez connu jusqu'alors. Vous partez avec un sujet de thèse, mais il s'agit d'un sujet de recherche... il est donc possible que vous deviez le modifier en cours pour diverses raisons comme l'actualité de votre domaine. Imaginez que vous tentiez de résoudre un problème très difficile avec une approche. Mais, au cours de votre deuxième année de thèse, quelqu'un publie son résultat, ou démontre que vous ne pourrez jamais le faire de cette manière.
Un autre point important, il va falloir apprendre à gérer votre temps. Il ne s'agit pas seulement de savoir comment vous vous organiserez pour finir un gros projet scolaire en conciliant toutes vos autres activités... Il s'agit de savoir comment vous devrez organiser votre vie pendant 3 ou 4 ans afin d'atteindre un objectif.

Quelles sont les compétences requises ?

En règle générale, tout doctorant se doit d'avoir un niveau d'anglais suffisant pour savoir lire des publications scientifiques rapidement, pour rédiger de nombreuses propositions d'articles afin de faire connaître ses résultats, et pour présenter ses travaux de manière compréhensible lors de conférences internationales.
Ensuite, en informatique et surtout en théorie des langages, un doctorant se doit d'avoir un bon niveau d'algèbre, linéaire ou non, afin de pouvoir réaliser toutes les démonstrations sous-jacentes à ses travaux. Dans le domaine de la vérification de systèmes, on peut même aller jusqu'à dire que la maîtrise des concepts mathématiques nécessaires à la construction de son modèle est parfois même ce qui fait la différence entre deux travaux a priori équivalents.

Contrairement à ceux que beaucoup pensent souvent, en règle générale un doctorant en informatique n'est pas obligé d'être un génie du code connaissant une vingtaine de langages de programmation parfaitement. La partie implantatoire arrive souvent comme une sorte de « récompense » de l'ensemble du travail accompli précédemment... Inutile de passer plusieurs heures (jours) à réaliser un prototype si l'on n'est pas quasiment sûr qu'il apportera quelque chose de nouveau de manière performante. Notre travail n'est pas de réinventer la roue. Réutiliser des programmes déjà testés par d'autres peut nous faire gagner un temps précieux, et nous permettre de nous concentrer sur le prototypage des parties réellement importantes de nos travaux.

Quel intérêt peut-on trouver dans la vérification de système ?

Tout le monde se souvient du fiasco que fut le lancement de la première fusée Ariane 5. Mais beaucoup ignorent que ce feu d'artifice a été causé par une petite erreur dans le logiciel informatique embarqué dans la fusée.
Ces bogues arrivent souvent lors de la réalisation d'un projet informatique, avec des conséquences fort heureusement bien moins graves la plupart du temps. Ils peuvent avoir des causes très variées, de la simple erreur d'inattention d'un programmeur à la faute de conception. Et suivant le domaine d'utilisation ils peuvent engendrer des désastres aussi bien sur le plan économique (réparation, mise à jour des versions déjà vendues), que sur le plan humain (imaginez que l'ordinateur qui contrôle la navigation de votre Airbus s'arrête en plein vol).

Depuis de nombreuses années, les domaines où la sûreté de fonctionnement est primordiale, exige de plus en plus de leurs outils de développement. Ils ont déjà mis en place des règles de conception et de codage, des méthodologies de développement incluant plusieurs séries de tests poussés. Malgré toutes ces précautions, ils ne sont pas à l'abri d'une erreur. En effet, dans un cadre purement générale, nous sommes face à une question indécidable... Aucun ordinateur ne pourra certifier qu'il n'existe aucun bogue dans un système quelconque que nous lui demanderions d'analyser.
Il faut donc réussir à définir un cadre très précis dans lequel le système devra se trouver, éventuellement en restreignant les fonctionnalités disponibles lors de la conception et de la réalisation. Puis la description de ce système (matériel ou logiciel) sera analysée par un outil adapté au domaine étudié, qui devra alors être en mesure de vérifier qu'aucun des états dans lequel il pourrait se trouver n'entre en conflit avec ce qu'on attend de lui.

En quoi consiste mon travail de recherche ?

Le but final de ma thèse serait de modéliser et concevoir un outil de débogage formel pour systèmes logiciels ou matériels selon l'approche « debug as design ». Bien sûr la plupart des personnes, hors de notre domaine de recherche, se demanderont ce que peut bien cacher cette expression...
Pour faire simple, nous avons choisi d'analyser des descriptions de systèmes. Pour un logiciel nous pourrions étudier son code source, alors que pour un circuit électronique nous nous intéresserions à sa description dans un langage adapté (HDL).
L'outil réalisé serait un débogueur formel, c'est-à-dire un outil capable de détecter les erreurs et de signaler leur emplacement sans avoir besoin de faire fonctionner réellement le système étudié. Cette technique a l'avantage de ne pas dépendre des conditions dans lesquelles cela a été réalisé, et donc de pouvoir certifier le résultat dans tous les cas possibles.
Enfin, l'approche « debug as design » consiste à utiliser des outils de vérification très tôt dans le cycle de conception du système. Ainsi il signalerait rapidement les petits problèmes locaux, et fusionnerait ensuite tous les sous-résultats pour analyser un plus gros morceau, jusqu'à finir par avoir vérifié l'intégralité du système.

Pour en savoir plus

On the design of a Formal Debugger for System Architecture
Bruno Monsuez, Franck Védrine, Nicolas Vallée
DATIDS'08 (Crète – Juillet 2008)

How an "Incoherent Behavior" inside generic hardware component characterizes functional errors ?
Bruno Monsuez, Franck Védrine, Micaela Mayero, Nicolas Vallée
CISST'09 (Chine – Janvier 2009)

Beyond Hierarchical graphs : Fractal Hypergraphs
Bruno Monsuez, Nicolas Vallée, Franck Védrine
(en attente de publication)