Baron, M. De la conception à la construction d'application sûre (Rencontre Doctorale). In *Proceedings of 14° Conférence Francophone sur l'Interaction Homme-Machine (IHM'2002)* (27-29 novembre 2002, Poitiers), ACM Press, 2002, pp. 285-286

De la conception à la construction d'application sûre

Mickaël Baron

Laboratoire d'Informatique Scientifique et Industrielle
ENSMA, BP 40109, 86961 Futuroscope Chasseneuil
http://www.lisi.ensma.fr/ihm
baron@ensma.fr

RESUME
Notre travail de thèse consiste à étudier le développement des applications sûres par deux approches différentes dans le but de comparer les apports de chacune. Dans cet article, nous sommes restreints à la première partie de notre travail, concernant l'environnement SUIDT.

MOTS CLES : Programmation visuelle, modèle de tâche, méthodes formelles, systèmes basés sur modèles.

ABSTRACT
Our thesis work consists in studing the development of safe application by the way of two approaches in order to compare each one. In this paper, we only present the first part of our work, it is about SUIDT environment.

KEYWORDS : Model-Based Systems, Visual Programming, Task Model, Formal Methods.

INTRODUCTION

Le développement de logiciels de qualité nécessite l'utilisation de techniques très rigoureuses. Elles doivent assurer que les logiciels en développement satisfont les propriétés qui traduisent les exigences définies dans le cahier des charges. Pour développer des applications interactives sûres, deux approches peuvent aujourd'hui être mises en parallèle : les Systèmes Basés sur Modèles (Model-Based Systems) d'une part, et les approches formelles d'autre part. L'approche des systèmes basés sur modèles tire parti des spécifications de modèles (modèle de tâche, modèle fonctionnel…), pour construire et valider des applications interactives, comme le système Mobi-D [4]. Cependant, la généralisation de l'utilisation de ces systèmes est freinée par leur difficulté d'utilisation. En effet, cette dernière est réservée à des spécialistes en informatique, et nécessite généralement l'apprentissage de langages spécifiques. Ces approches semi-formelles n'ont pas la possibilité d'utiliser la puissance d'une sémantique formelle pour effectuer de la vérification et de la validation. Pour y remédier de nombreux travaux ont été menés dans le but d'introduire une démarche formelle dans le développement des systèmes interactifs [2]. Cependant, l'utilisation de ces méthodes ne s'est pas imposée car elles demeurent difficiles d'accès et n'ont pas encore franchies les test d'échelles. Les cas étudiés sont encore, au moins au plan interactif, très limités.

IHM 2002, November 26-29, 2002, Poitiers, France

L'objet de ce travail de thèse est l'étude du développement des applications sûres par deux approches différentes dans le but de comparer les apports de chacune.

Une première démarche vise à faire collaborer les approches « constructeur d'interface » (GUI-Builder) et « systèmes basés sur modèles » afin de définir une méthode et un environnement de construction d'applications interactives sûres s'appuyant sur un noyau fonctionnel formel. Cette partie est en cours de finalisation, elle est encadrée par Patrick Girard, professeur à l'Université de Poitiers. La seconde démarche est purement formelle. Elle s'intéresse à la modélisation et à la validation d'une application interactive tout en respectant les besoins de l'utilisateur. Cette partie vient de débuter ; elle est encadrée par Yamine Aït-Ameur, professeur à SUP-AERO, Toulouse.

Nous nous concentrons tout particulièrement dans cet article sur la description des travaux de la première partie de notre travail.

CONCEPTION D'APPLICATION SURE

Un premier travail a abouti au système GenBUILD [1]. Cet outil génère automatiquement une application interactive à partir de l'interface d'un noyau fonctionnel préalablement développé. L'originalité de la démarche réside dans le fait que cette application, générée automatiquement, est couplée à un constructeur d'interface. Celui-ci permet de réaliser une véritable application interactive tout en conservant le lien avec le noyau fonctionnel. Deux points limitent l'approche de GenBUILD. D'une part, en l'absence d'un modèle de tâche, le système ne peut garantir que les actions de l'utilisateur sur l'application peuvent être atteintes. D'autre part, la description du noyau fonctionnel s'appuie sur une sémantique pauvre.

Nous proposons actuellement une nouvelle démarche qui comble les limites de GenBUILD. Notre environnement interactif pour utilisateur final, appelé SUIDT (Safe User

Interface Design Tool) valide cette démarche. Il regroupe plusieurs constructeurs de modèle : un modèle fonctionnel, un modèle de tâche et un modèle d'interface. Il est destiné à deux types d'utilisateurs: un spécialiste des méthodes formelles pour le développement du noyau fonctionnel et un concepteur d'application pour l'interface.

Un noyau fonctionnel formel

La démarche de SUIDT s'appuie sur un noyau fonctionnel sûr, développé indépendamment de toute perspective d'interaction. Pour cela nous avons choisi comme formalisme une méthode basée sur modèle, où l'essentiel de la sémantique peut s'exprimer sous la forme d'invariants. La méthode utilisée dans GenBUILD s'adapte directement à un noyau fonctionnel formel et nous l'avons repris dans SUIDT. La méthode B a été choisi pour cela. L'analyse des spécifications du noyau fonctionnel permet de générer automatiquement une interface de manipulation de ce même noyau.

L'avantage de cette approche est de disposer d'un noyau fonctionnel que l'on peut tester et utiliser pendant le développement de l'application. Cependant, dans le cas d'une véritable application, l'appel des fonctions du noyau doit suivre un (ou des) scénario-(i) précis. C'est la raison pour laquelle nous avons inclus dans l'approche de SUIDT un modèle de tâche.

Un modèle de tâche

Nous avons introduit la notion de modèle de tâche dans notre approche, afin de permettre de décrire au mieux l'application du point de vue de l'utilisateur. C'est à partir de ce modèle de tâche, et dans le respect des contraintes du noyau fonctionnel, que le concepteur d'applications interactives sera à-même de construire son application. Nous avons recherché une méthode de description de tâches qui nous permette d'exprimer une analyse de tâche au sein de notre outil. Notre choix s'est porté sur CTT (ConcurTaskTrees) [3], et a été motivé par le fait que ce formalisme utilise une notation graphique, à sémantique parfaitement définie et dispose d'un outil CTTE (ConcurTaskTrees Environment) qui s'avère relativement ouvert sur l'extérieur (possibilité d'exporter des modèles en XML).

Nous distinguons deux niveaux distincts de modèle de tâche. Le niveau appelé « modèle de tâche abstrait » où aucun choix de technique d'interaction ne sera autorisé et le niveau appelé « modèle de tâche concret » destiné uniquement à concrétiser en terme d'interactions les tâches abstraites. Le modèle de tâche concret s'appuie complètement sur le modèle de tâche abstrait, et consiste en un raffinement des tâches abstraites. Afin de définir nos deux niveaux de modèle de tâche, nous avons utilisé CTT en le restreignant. Cependant, nous avons systématiquement évité de modifier la sémantique de CTT, afin de conserver ses propriétés formelles. La dernière partie du travail consiste à coupler notre outil à un GUI-Builder permettant de concrétiser l'application. Cette partie est actuellement en cours de réalisation.

DISCUSSION

L'outil SUIDT est un véritable environnement qui regroupe un animateur de noyau fonctionnel, un outil qui permet de construire le modèle de tâche et un GUI-Builder. Notre outil est proche de CTTE car il offre aussi une édition et une visualisation graphique du modèle. Mais l'outil CTTE ne manipule que le modèle de tâche, et il n'y a pas de liaison avec d'autres modèles. SUIDT à l'inverse regroupe un noyau fonctionnel, un modèle de tâche et un modèle d'interface tout comme les systèmes basés sur modèle. L'inconvénient majeur de l'absence de liaison entre modèles se situe au niveau de la validation du modèle de tâche. L'outil de validation de CTTE contrôle le modèle de tâche de toutes sortes d'incohérences mais n'assure pas comme celui de SUIDT que les appels entre chaque modèle sont cohérents.

Nous avons montré par une méthode et un environnement qu'il était possible de concevoir une application interactive sûre en s'appuyant sur des modèles sémantiquement définis et des techniques de programmation visuelle. La partie semi-formelle qui vise à faire collaborer les approches « constructeur d'interface » et « systèmes basés sur modèles » se termine avec le développement de l'outil SUIDT. Outre l'industrialisation de SUIDT (échange possible entre CTTE et SUIDT) nous allons nous intéresser dés à présent à une approche purement formelle.

BIBLIOGRAPHIE.

1. Baron, M. et Girard, P. Bringing Robustness to End-User Programming. In *Proceedings of 2001 IEEE Symposia on Human-Centric Computing Languages and Environments* (September 5-7 2001, Stresa, Italy), Entergraphica, 2001, pp. 142-149.
2. Jambon, F., Brun, P. et Aït-Ameur, Y. Spécifications des systèmes interactifs (chapitre 6). Kolski, C. (Ed.). In *Analyse et conception de l'I.H.M. / Interaction Homme-Machine pour les S.I. vol.1*, Hermès Science, Paris, France, 2001, pp. 175-206.
3. Paternò, F. *Model-Based Design and Evaluation of Interactive Applications*. Springer, 2001.
4. Puerta, A. et Eisenstein, J. Interactively Mapping Task Model to Interfaces in Mobi-D. In *Proceedings of Eurographics Workshop on Design, Specification and Validation of Interactive Systems (DSV-IS'98)* (3-5 June, Abingdon, UK), 1998, pp. 261-274.