Un langage formalisable pour la définition d'applications interactives embarquées et critiques

Vincent Lecrubier
ONERA
2 Avenue Edouard Belin
31400, Toulouse, France
Vincent.Lecrubier@onera.fr

Bruno d'Ausbourg
ONERA
2 Avenue Edouard Belin
31400, Toulouse, France
Bruno.d.Ausbourg@onera.fr

Yamine Aït-Ameur
INPT/ENSEEIHT
2 rue Charles Camichel
31000, Toulouse, France
yamine@enseeiht.fr

RÉSUMÉ
Cet article décrit et utilise le format de mise en page à respecter pour les soumissions à IHM'13. Après acceptation, ces soumissions seront publiées dans les actes d'IHM'13, ainsi que sur l'ACM Digital Library, sauf les communications informelles qui seront publiées dans un volume annexe séparé. Comme certains détails ont changé par rapport aux années précédentes, nous vous prions de bien lire ce document même si vous avez déjà soumis à IHM auparavant.

Mots Clés
Format; instructions; qualité; actes de conférence; les mots-clés doivent être séparés par des points-virgules.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous. Voir `http://www.acm.org/about/class/1998/` pour la liste complète des catégories ACM.

INTRODUCTION

Méthodes et techniques de descriptions formelles ont été, depuis longtemps, appliquées au domaine des des systèmes critiques pour des besoins de sûreté. C'est particulièrement le cas des systèmes aéronautiques dont le fonctionnement peut mettre en jeu la vie de passagers. De nombreux travaux de recherche se sont attachés à élaborer et proposer des moyens pour concevoir, programmer et vérifier les fonctionnalités de systèmes critiques en particulier aéronautiques. Ces systèmes et les processus qui encadrent leur développement sont par ailleurs l'objet d'étapes de certification qui établissent que le produit et le développement qui a conduit à le construire sont en conformité avec des standards établissant des règles précises les définissant, tels la DO178C [1] pour le logiciel aéronautique embarqué.

Les systèmes d'interaction homme-machine (IHM) n'ont bénéficié ni de la même attention ni du même effort de recherche. Pourtant des systèmes d'interaction mettant en oeuvre des capacités électroniques et numériques hautement sophistiquées ont fait leur apparition au sein de cockpits d'avions de nouvelle génération. Ces systèmes se fondent sur des logiciels à la fois volumineux et complexes et doivent répondre à des contraintes sévères en matière d'utilisabilité, de sécurité ou de sûreté. Ces logiciels mettent en oeuvre des applications dont le comportement, compte tenu de leur criticité, doit se conformer avec un niveau très élevé d'assurance à ce qui est attendu du système. En effet une erreur dans l'un des composants d'interaction logiciels peut conduire à une erreur humaine ou système dont les effets peuvent se révéler catastrophiques.

Le récent rapport du Bureau d'Enquêtes et d'Analyses [9] sur le crash de l'Airbus A330 d'Air France lors du vol AF447 Rio-Paris pointe un réel dysfonctionnement du système d'interaction, et plus particulièrement du directeur de vol, ayant affiché des indications qui n'auraient pas dû l'être et ayant ainsi conduit le pilote à cabrer son appareil alors que ce dernier était déjà en train de décrocher.

Le coeur du problème réside essentiellement dans le fait que les processus standards de développement des systèmes aéronautiques critiques se fondent la plupart du temps sur des cycles linéaires (du type des cycles en V ou de ses variantes) qui s'avèrent peu ou mal adaptés à la conception et au développement des logiciels d'interaction qui requièrent des cycles d'itérations impliquant les usagers et mêlant opérations de test et phases de conception dans les processus mis en oeuvre. Une autre difficulté, propre au développement des systèmes d'interaction, est imputable à la multiplicité des communautés techniques et scientifiques intervenant dans le champ de la définition des systèmes d'IHM. Manquent à ces communautés diverses un langage et des notations qui soient, sinon communes, du moins intelligibles par toutes. Dès lors le traitement de la sûreté et de la sécurité de ces systèmes d'interaction critiques ne fait appel à aucune méthode ni aucun outil spécifiques. Dans ce contexte, les processus de validation de ces systèmes demeurent pauvres car ils se limitent pratiquement à des phases intensives de test et d'évaluation en fin de processus de développement, qui plus est sans disposer de réelle référence, en particulier formelle, vis à vis de laquelle confronter le système développé.

Ce papier suggère d'introduire cette référence formelle lors des étapes amont des processus de développement des systèmes d'IHM critiques en proposant un langage,

formalisable, qui permette la description des applications interactives, tant du point de vue de leur présentation que de leur comportement, ainsi que l'expression d' exigences des concepteurs concernant l'interaction encodée au sein de ces applications. Ce langage veut être suffisamment abstrait et convivial pour être compris et utilisables par tous les participants à la conception et au développement de l'IHM. Il veut également être, sinon formel, du moins formalisable de telle sorte que qu'un texte rédigé dans ce langage puisse être soumis à des opérations automatiques ou semi-automatiques de vérifications et de preuves formelles, de génération de tests pertinents du système à développer, et de génération de code plus sûr. De manière à accroître l'assurance que le logiciel développé se comporte bien comme prévu.

TRAVAUX CONNEXES

La formalisation des systèmes d'interaction a pourtant fait l'objet de travaux importants lors de ces dernières années: bon nombre de modèles, de langages ou de méthodes formels ont vu le jour pour aider à spécifier, concevoir, vérifier et implanter ces systèmes.

Une première classe de modèles basés sur les systèmes de transitions et plus particulièrement sur les extensions d'automates d'états finis et de réseaux de Petri ont utilisé les automates d'états finis pour décrire des comportements interactifs de base. De manière à circonvenir les problèmes liés à l'explosion combinatoire de ces modèles de nombreuses extensions ont été proposées. Les machines d'états hiérarchiques ou les statecharts décrivent des automates composables facilitant la spécification de systèmes complexes et permettant la réutilisation des modèles. Les automates à états finis peuvent être utilisés dans le cadre d'outils de programmation [10, 11] ou peuvent être associés à d'autres formalismes pour décrire d'autres aspects de l'interaction. En [14] les automates modélisent les apsects de l'interaction liés à des entrées discrètes tandis que les flots de données représentent les entrées continues.

Le modèle de capteurs formels [2] utilise les réseaux de Petri de haut niveau [15] pour décrire les transdormations successives d"événements d'entrée. Les réseaux de Petri ont été utilisés par plusieurs méthodes et techniques pour décrire les systèmes d'IHM. Le formalisme ICO [16, 17] se fonde sur une approche orientée objet pour modéliser les composants statiques d'un système d'interaction et sur les réseaux de Petri pour en décrire la dynamicité : PetShop [8, 19] est l'outil basé sur ICO.

D'autres descriptions formelles des systèmes interactifs se fondent sur la notion d'interacteurs. Les interacteurs peuvent se concevoir comme des entités autonomes et communicantes interagissant entre elles, de manière interne au système d'interaction, ou interagissant de façon externe avec le système interfacé ou l'usager par le biais d'événements (ou de stimuli). Les deux premiers modèles d'interacteurs développés le furent dans le cadre du projet AMODEUS. Le modèle de York [13] fournit un cadre pour la description de systèmes d'interaction à l'aide de notations orientées modèles telles que Z ou VDM. Le modèle du CNUCE [18] utilise quant à lui LOTOS. Des interacteurs formels ont également été décrits en utilisant le langage synchrone Lustre : [6] montre comment de tels interacteurs peuvent être dérivés d'une description informelle du système d'interaction. Ces interacteurs formels peuvent être composés pour constituer un modèle formel de l'IHM et peuvent être utilisés pour vérifier le système conçu ou générer des tests [7] et pour, plus généralement, valider le système d'interaction développé [12].

[3, 5] introduisent une approche modulaire basée sur l'usage de la méthode B. Les spécifications d'un système interactif opérationnel y sont formalisées en assurant qu'un certain nombre de propriétés ergonomiques sont préservées par le système. Diverses formes d'interaction ont ainsi été formalisées dans le cadre de ces travaux en permettant la vérification de nombreuses propriétés de ces systèmes telles leur observabilité, leur honnêteté ou leur robustesse. Le travail présenté en [4] montre la description et la preuve, à l'aide de B, de toutes les étapes de raffinement d'un modèle abstrait de système jusqu'à son codage concret.

Le grand mérite de ces travaux est ainsi d'avoir démontré que l'interaction peut être formalisée et est donc formalisable. Toutefois ces approches construisent leurs modèles directement dans les formalismes cibles. Cela suppose la maîtrise de leurs notations. Or les différents intervenants dans le cadre de la définition de ces systèmes ne sont jamais des praticiens de ces formalismes qui, dès lors, demeurent inutiles et sans grand intérêt pour eux. Il était donc nécessaire de construire un langage utilisable ayant toutefois la capacité d'offrir les mêmes services.

LE BESOIN AUQUEL NOUS VOULONS RÉPONDRE

Le langage proposé vise donc à répondre à différents besoins exprimés par beaucoup d'acteurs dans le domaine de IHM embarquées critiques.

Le premier de ces besoins est de pouvoir vérifier certaines proprietés de sûreté, le plus tôt possible dans le cycle de développement. A l'heure actuelle, les propriétés de sûreté ne sont vérifiées dans le meilleur des cas qu'après une certaine concrétisation du concept d'IHM, et dans le pire des cas qu'après l'implémentation complète de l'IHM, lors d'une campagne de tests. Certaines erreurs n'apparaissent que lors de la validation finale, faisant perdre beaucoup de temps et d'argent, alors qu'elles auraient pu être corrigées dès les premières étapes de la spécification. Il manque donc un moyen de vérifier formellement des proprietés de sûreté dès les premières phases d'un projet, quand celui ci est encore abstrait.

Le second besoin provient de l'enrichissement actuel du monde de l'IHM. Ce monde vit actuellement une phase de grande diversification, et de nombreux horizons s'offrent aux concepteurs en termes d'interaction. Cependant, en l'absence d'outils permettant de travailler sur les aspects conceptuels des IHM, les concepteurs se mettent rapidement à raisonner en termes d'implémentation concrète, en se basant sur des notions qu'ils connaissent déjà. Ainsi, ils passent trop rapidement du besoin à la solution, négligeant d'envisager les solutions les plus récentes et plus adaptées à leur besoin. Il manque donc un moyen

de travailler efficacement sur les aspects conceptuels des IHMs, permettant de se poser les bonnes question en termes d'interaction.

Le troisième besoin est d'ordre humain. Les différents acteurs travaillant à la conception d'une IHM ont tous des métiers différents. On trouve pêle mêle, des designers, des programmeurs, des ergonomes, des ingénieurs systèmes, ainsi que des spécialistes en facteurs humains ou en architecture logicielle. Ces différents métiers ont des cultures variées, et donc assez peu de points communs. Ces barrières rendent parfois la communication peu efficace, compliquant le processus de conception des IHM. Il est donc important d'offrir un langage commun à ces différents acteurs afin de fluidifier leur collaboration.

Enfin, le dernier besoin, lié au précédent, est dû à l'absence de lien formel entre les différents artefacts générés par les différents acteurs. La conception d'une IHM critique passe généralement par la réalisation de produits intermédiaires : documents de spécification, documentation de cas de tests, différents prototypes, code d'implémentation... Ces différents documents ont un lien assez laxe et informel entre eux, ce qui pose des problèmes de traçabilité et de synchronisation entre les différentes étapes du processus. Il y a donc clairement besoin d'un document pivot entre ces différents domaines.

Le langage proposé est développé dans le but de répondre au mieux à ces différents besoins. Ces différents besoin sont parfois contradictoires. Par exemple, pour pouvoir être compris par tous les acteurs, le langage doit être assez simple, ce qui s'oppose au besoin d'expressivité, nécéssaire pour pouvoir vérifier des proprietés formelles intéressantes. Notre travail consiste donc à trouver la solution présentant le meilleur compromis face à tous ces besoins.

LE LANGAGE

Le langage proposé s'articule autour de la notion d'Interacteur. Les interacteurs sont composés : un interacteur peut être composé de sous-interacteurs. Ainsi, une IHM complète sera représentée sous la forme d'un interacteur, composé de ses sous éléments : fenêtres, contrôles, sons... qui sont eux même des interacteurs.

Le langage vise à modéliser une interface dite abstraite. Cela signifie que l'on ne modelisera, au moins dans un premier temps, que les aspects conceptuels de l'IHM, sans s'occuper de leur implémentation réelle sur le produit fini. Ainsi, l'on pourra abstraire différents types d'entités sous un même concept, ce qui simplifie le langage et son utilisation. Par exemple, les bien connus boutons à deux états (ToggleButton) et case à cocher (CheckBox) ont une même fonction : permettre à l'utilisateur de communiquer une valeur booléenne à la machine; ils sont donc regroupés sous le concept d'entrée booléenne (BooleanInput). Tous les éléments d'IHM sont ainsi regroupés dans différentes catégories représentant les briques de bases de l'interaction homme-machine.

Les types d'interacteurs de base du langage sont ainsi en nombre relativement réduit, tout en permettant de prendre en compte l'intégralité des interactions homme-machine. On trouvera des entrées et sorties de textes, réels, entiers, booléens, choix de sous ensembles parmi un ensemble, ainsi que de déclenchement d'évenements. La nature du langage permet de combiner ces briques de base afin de définir des interactions plus complexes.

On l'a vu, un interacteur se compose de sous interacteurs. Ceci permet de représenter la structure statique des IHM. Afin d'exprimer le dynamisme des IHM, un interacteur sera aussi composé d'une partie exprimant sont comportement. Le comportement des interacteurs sera exprimé à l'aide d'une machine à états. L'interacteur comprendra ainsi un ensemble de variables d'état et un ensemble de règles définissant l'évolution des ces variables d'état, en fonction des évenements reçus par l'interacteur par le biais de ses sous-interacteurs.

Ces règles d'évolution de l'état de l'interacteur permettent d'exprimer plusieurs types de comportements.

Un premier type de comportement représente l'état initial de l'interacteur et des sous interacteurs. Il permet d'exprimer des proprietés qui seront vraies à l'initialisation de l'interacteur, sans forcément le rester par la suite. Par exemple, à l'ouverture d'une page web, la vue est placée tout en haut de la page.

Un autre type de comportement est défini comme un invariant. Il représente des proprietés qui resteront vraies à tout instant lors de l'exécution. Ainsi, l'heure affichée dans la barre d'état d'un smart phone, est à tout moment égale à l'heure telle que se le représente le système.

Un dernier type de comportement est défini comme la réponse à un évenement reçu. Il spécifie que si l'interacteur reçoit un certain évenement, et que certaines conditions (garde de la transition) sont réunies, alors un changement d'état aura lieu. Par exemple, si l'on clique sur le bouton "quitter", la fenêtre se fermera.

BIBLIOGRAPHIE

1. DO178C. Software Consideration in Airborn Systems and Equipment Certification, release c, 2012. RTCA,Inc.
2. Accot, J., Chatty, S., Maury, S., and Palanque, P. A. Formal transducers: Models of devices and building bricks for the design of highly interactive systems. In *DSV-IS*, M. D. Harrison and J. C. Torres, Eds., Springer (1997), 143–159.
3. Ait-ameur, Y., Girard, P., and Jambon, F. A uniform approach for specification and design of interactive systems: the b method. In *in Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSVIS'98), Vol. Proceedings (Eds, Markopoulos* (1998), 333–352.
4. Ameur, Y. A. Cooperation of formal methods in an engineering based software development process. In *Proceedings of the Second International Conference on Integrated Formal Methods*, IFM '00, Springer-Verlag (London, UK, UK, 2000), 136–155.
5. Ameur, Y. A., Girard, P., and Jambon, F. Using the b formal approach for incremental specification design of interactiv systems. In *EHCI* (1998), 91–109.
6. Ausbourg (d'), B., Durrieu, G., and Roché, P. Deriving a formal model of an interactive system from its UIL description in order to verify and to test its behaviour. In *Proceedings of the Eurographics Workshop DSV-IS'96* (Namur, Belgium, June 1996).
7. Ausbourg(d'), B. Using Model Checking for the Automatic Validation of User Interfaces Systems. In *Design, Specification and Verification of Interactive Systems '98*, P. Markopoulos and P. Johnson, Eds., Eurographics, Springer (June 1998).

8. Bastide, R., Navarre, D., and Palanque, P. A model-based tool for interactive prototyping of highly interactive applications. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, ACM (New York, NY, USA, 2002), 516–517.

9. BEA. Rapport final sur l'accident survenu le 1er juin 2009 à l'Airbus A330-203 immatriculé F-GZCP exploité par Air France, vol AF 447 Rio de Janeiro - Paris. Tech. rep., Direction Générale de l'Aviation Civile, Juillet 2012. **`http://www.bea.aero/docspa/2009/f-cp090601/pdf/f-cp090601.pdf`**.

10. Blanch, R. Programmer l'interaction avec des machines à états hiérarchiques. In *Actes de la 14ème conférence francophone sur l'Interaction Homme-Machine (IHM 2002)* (2002), 129–136.

11. Blanch, R., and Beaudouin-Lafon, M. Programming rich interactions using the hierarchical state machine toolkit. In *Proceedings of the working conference on Advanced Visual Interfaces (AVI 2006)* (2006), 51–58.

12. d'Ausbourg, B., Seguin, C., Durrieu, G., and Roché, P. Helping the automated validation process of user interfaces systems. In *ICSE*, K. Torii, K. Futatsugi, and R. A. Kemmerer, Eds., IEEE Computer Society (1998), 219–228.

13. Duke, D. J., and Harrison, M. D. Abstract Interaction Objects. *Computer Graphics Forum 12*, 3 (1993), 25–36.

14. Jacob, R. J. K., Deligiannidis, L., and Morrison, S. A software model and specification language for non-wimp user interfaces. *ACM Transactions on Computer-Human Interaction 6* (1999), 1–46.

15. Jensen, K. *Coloured Petri nets: basic concepts, analysis methods and practical use, vol. 2*. Springer-Verlag, London, UK, UK, 1995.

16. Palanque, P., and Bastide, R. Interactive cooperative objects : an object-oriented formalism based on petri nets for user interface design. In *Proceedings of the IEEE Conference on System Man and Cybernetics*, Elsevier Science Publisher (October 1993).

17. Palanque, P., and Bastide, R. Synergistic modelling of tasks, users and systems using formal specification techniques. *Interacting with Computers 9*, 2 (1997), 129–153.

18. Paternò, F., and Faconti, G. On the use of LOTOS to describe graphical interaction. In *Proceedings of the HCI'92 Conference on People and Computers* (1992), 155–173.

19. Schyn, A., Navarre, D., Palanque, P., and Porcher Nedel, L. Description Formelle d'une Technique d'Interaction Multimodale dans une Application de Réalité Virtuelle Immersive . In *IHM'2003: 15th French Speaking conference on human-computer interaction, Caen, France, 24/11/2003-28/11/2003*, ACM Press (novembre 2003), 150–157.