Un langage pour la spécification des interfaces homme-machine critiques

Lecrubier

Vincent

3ème année

DTIM

Mél. : vincent.lecrubier@gmail.com Tél. : 0679709196

Directeur(s) de thèse :
Ausbourg(d'), Bruno, ONERA, DTIM
Aït-Ameur, Yamine, ENSEEIHT

Site : Toulouse

Thèse financée par : ONERA

Mots clés : Interaction homme-machine, système critique, logiciel embarqué, spécification, vérification, validation, langage dédié, langage formel

Contexte

Le secteur aéronautique est depuis longtemps à la pointe dans le domaine du logiciel critique. Ainsi, les catastrophes causés par des défaillances logicielles sont devenus rarissimes. A l'inverse, des « erreurs humaines » sont très souvent citées parmi les causes d'accidents. Mais dans la plupart des cas, il faudrait plutôt parler de malentendu entre l'homme et la machine. Le danger survient lorsque l'homme se fait une représentation mentale qui ne correspond pas à l'état réel du système. La source de l'erreur se trouve donc bien souvent entre l'homme et la machine, autrement dit au niveau de l'interface homme-machine (IHM).

Cette thèse se trouve à la confluence de deux domaines : L'interaction homme-machine d'une part, et le logiciel critique d'autre part. Ces deux domaines sont complémentaires dès lors que l'on s'intéresse aux IHM critiques, mais ils sont néanmoins en opposition sur beaucoup d'aspects.

Dans le domaine des systèmes critiques, les projets ont généralement des cycles de développement assez longs et couteux, avec des spécifications strictes et, de plus en plus, formalisées mathématiquement. Le traditionnel cycle en V est encore un modèle assez présent, avec des phases de conception et de vérification clairement séparées.

Le domaine de l'interaction homme-machine connaît depuis quelques années un développement considérable. En effet, le succès de produits informatiques sont de plus en plus conditionnés par la qualité de leur interface utilisateur. En particulier, les retours des utilisateurs sur le produit est pris en compte très en amont, et les cycles de développement les plus efficaces dans ce domaine sont très éloignés du cycle en V traditionnel.

Le rapport entre la croissance de la puissance de calcul des machines et la finitude des capacités de compréhension de leurs utilisateurs humains est la cause principale de la prise d'importance des aspects interactifs des systèmes. D'autre part, la dernière décennie a vu une explosion des modalités d'interaction : Interfaces tactiles, commande gestuelle, commande vocale, interfaces haptiques. Ces nouvelles modalités d'interaction, bien que simplifiant l'IHM du point de vue de l'utilisateur, les rendent en réalité plus complexes à développer.

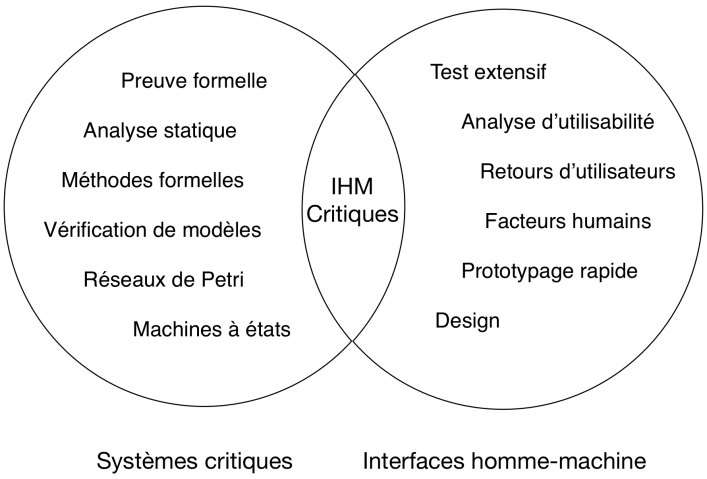

Figure 1 : Deux domaines en opposition

Objectifs Scientifiques

Concevoir un langage permettant la spécification, la vérification et la validation des interfaces homme-machine critiques. La principale contrainte à respecter consiste à situer le travail au juste milieu entre les domaines de l'interaction homme-machine et du logiciel embarqué, afin de réconcilier les deux cultures.

Ainsi, le langage devra répondre aux besoins des acteurs traditionnels dans le domaine de l'IHM : spécialistes en facteurs humains, ergonomes, designers, codeurs spécialisés en IHM, utilisateurs finaux, etc. Il faudra donc un langage suffisament simple pour être compris par de nombreux acteurs non formés à la programmation, mais permettant cependant d'exprimer et de spécifier les notions importantes en matière d'IHM, tout en facilitant le travail des spécificateurs.

D'un autre côté, le langage devra autoriser l'application de méthodes de vérification et de validation permettant de répondre aux contraintes exigeantes du domaine des systèmes embarqués critiques. En particulier, le langage devra permettre de formaliser de manière non ambiguë le comportement des IHM et de vérifier certaines propriétés de sûreté et de performance.

Démarche et déroulement

Dans un premier temps, un état de l'art des travaux scientifiques concernant la formalisation de l'interaction homme machine a été effectué. Cet état de l'art a permis de mettre au jour quelques approches expérimentales dans le domaine (voir bibliographie pour quelques exemples). Aucune des approches identifiées ne répond à l'intégralité des objectifs scientifiques de cette thèse, c'est pourquoi la suite du travail s'est orienté vers la définition d'un nouveau langage, plutôt que l'amélioration d'outils existants.

Une première version du langage (appelé L comme Langage) a été développée en s'inspirant des meilleurs éléments de l'état de l'art scientifique. Cette première version a ensuite été modifiée itérativement grâce au retours lors des premières présentation du langage, tout d'abord avec mes directeurs de thèse, puis lors de diverses présentations lors de workshops, réunions et séminaires.

Différentes cas de test de la méthode ont été déroulées, afin de valider l'utilisation du langage sur des cas concrets, et en évaluer l'intérêt du point de vue de l'utilisateur du langage, c'est à dire du spécificateur d'IHM. Ces différentes applications ont permis de mettre au jour certaines limitations qui ont semblé compromettre l'intérêt du langage dans des cas réels. Des doutes existaient quant à l'utilisabilité du langage dans un contexte industriel, et non pas seulement scientifique et académique.

Afin de contourner ces obstacles, il a fallu adopter le point de vue de l'utilisateur du langage. C'est pourquoi une deuxième phase d'exploration de l'état de l'art a été lancée, cette fois orientée vers les solutions de conception d'IHM éprouvées et utilisées actuellement dans l'industrie. Cette phase a permis de mieux situer le langage, d'en dessiner plus clairement les contours ainsi que les avancées qu'il représente. En particulier, les

approches de développement d'IHM utilisées actuellement offrent des capacités d'abstraction, de vérification et de validation extrêmement limitées.

En offrant une vision plus complète du panorama du côté des utilisateurs, cette deuxième étude de l'état de l'art a permis d'ouvrir de nouvelles portes pour le développement du langage. Une deuxième version du langage beaucoup plus applicable a donc été développée et testée dans quelques cas d'application, et est en cours de formalisation.

Le principal outil utilisé pour formaliser le langage et spécifier les transformations de modèle et génération de code s'appelle Xtext. Il s'agit à la fois d'un framework basé sur la plate forme Eclipse, et d'un langage dédié au développement de langages dédiés.

Principaux résultats obtenus

Le résultat principal de ce travail est le langage lui même, avec sa syntaxe mais surtout sa sémantique.

Cette sémantique permet de représenter des interfaces dites abstraites. Une interface abstraite est une interface faisant abstraction des détails d'implémentation, et se concentrant sur les éléments ayant un vrai sens en matière d'interaction. Il ne sera donc pas question de pixels, de gestionnaires de fenêtres et de pilotes logiciels. Nous libérons de ces questions, et nous nous baserons sur trois notions principales plus abstraites :

- La première est la notion d'acteur. Il s'agit d'entités extérieures à l'IHM mais en interaction avec celle-ci. Ils tombent généralement dans deux catégories : homme ou machine. Ainsi, une IHM simple est définie en déclarant deux acteurs : un homme (l'utilisateur), et une machine (le système utilisé). Cependant, le cas général est plus complexe et fait intervenir plusieurs hommes et plusieurs machines. Les acteurs sont extérieurs à l'IHM et sont donc simplement déclarés dans le langage L, et non pas modélisés en détail. Les acteurs sont représentés par des bonhommes sur la Figure 1.
- La deuxième est la notion de donnée. Il s'agit simplement de la déclaration de types de données qui formeront les flux d'information circulant entre les différents acteurs via l'IHM. Cette notion est très familière aux informaticiens et ne sera donc pas développée plus en détail ici. Pour donner un exemple très simple, une donnée peut être un texte ou un nombre. Les données sont représentés par des flèches sur la Figure 1.
- Enfin, le cœur du sujet est compris dans la notion d'interacteur. Une IHM est définie comme un ensemble d'interacteurs composés de sous-interacteurs. Nous obtenons donc une structure arborescente d'interacteurs. Par exemple, l'IHM est un interacteur, qui peut être composée de boutons, de zones de textes ou autres, qui sont eux-mêmes des interacteurs. Cet ensemble d'interacteurs organisé hiérarchiquement, forme l'IHM dans son ensemble. Les interacteurs sont représentés par des rectangles arrondis sur la Figure 1.

Chaque interacteur est composé de trois types d'objets :

- Sa composition en sous-interacteurs s'ils existent. Cette décomposition est signalée par les flèches avec losange sur la Figure 1.
- La déclaration des différents signaux qui le parcourent. Ces signaux peuvent être des flux continus ou des évènements ponctuels. Ces signaux sont associés à des types de données. De plus, ces signaux peuvent soit rester à l'intérieur de l'interacteur (dans ce cas ils agiront comme des sortes de variables d'état), soit être communiqués à d'autres entités (interacteurs ou acteurs). Les signaux sont représentés par des rectangles droits sur la Figure 1.
- La déclaration des différents comportements de l'interacteur. Ces comportements sont spécifiés sous la forme d'une cause qui déclenchera le comportement, et d'effets qui sont des actions à effectuer sur les signaux lorsque le comportement se déclenche. Les comportements sont représentés par des cercles sur la Figure 1.

Ces notions permettent de répondre aux objectifs du langage et ont été testées sur de nombreux petits cas de test.

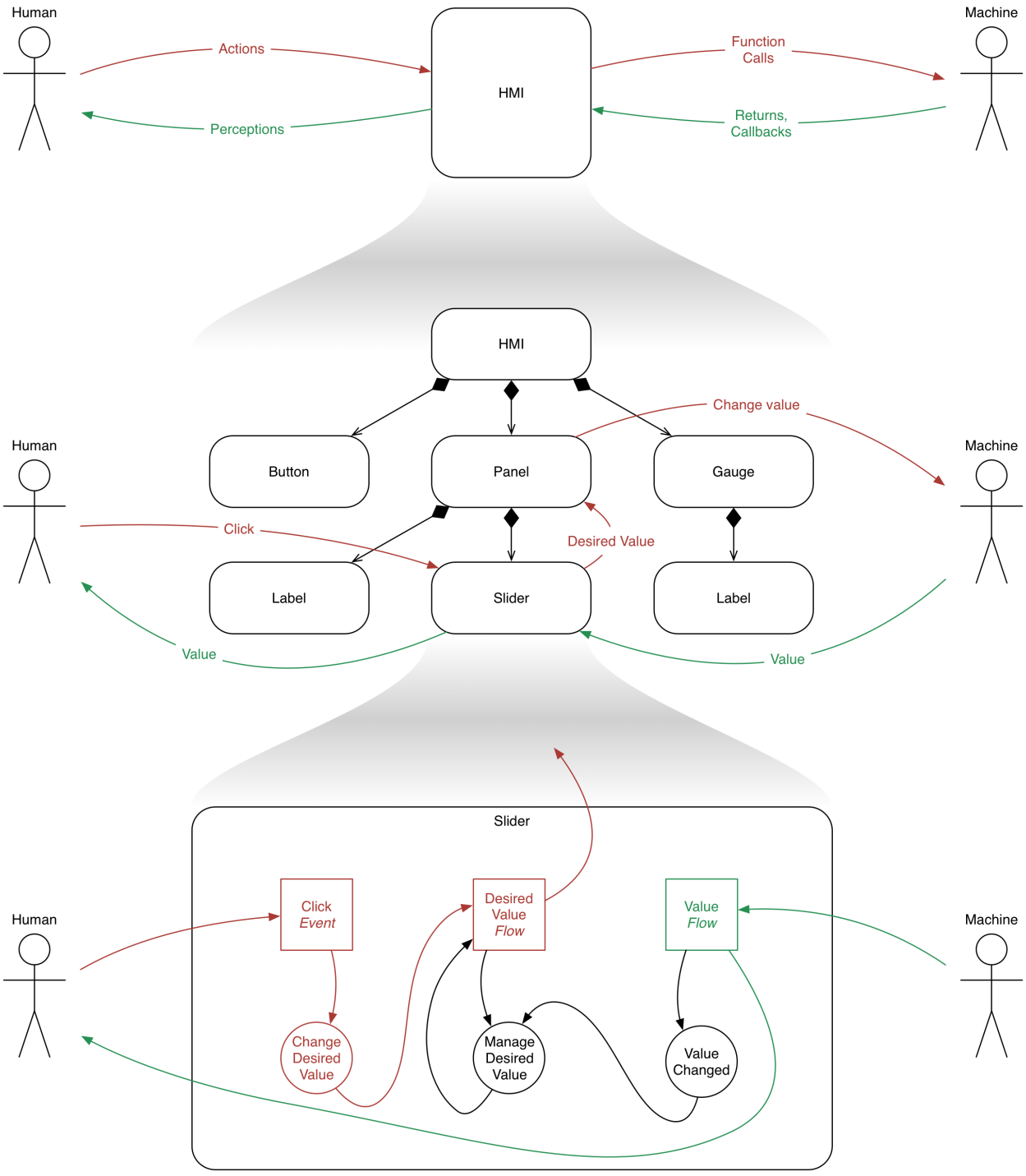

Figure 2 : Trois niveaux de détail des concepts du langage, sur un exemple simple.

Perspectives

La formalisation complète du langage et la génération de code d'exécution (En C ou en Java par exemple) et de vérification (En Promena ou en Event B par exemple) sont les principales les perspective à court terme, d'ici la fin de la thèse. Une autre piste à creuser est l'intégration de l'expression des propriétés à vérifier, directement dans le code L et dans le modèle.

A long terme, les perspectives consistent à poursuivre le développement du framework L afin soit de pouvoir générer du code certifié ARINC-661 et DO-178 d'une part, soit d'en augmenter les capacités afin de pouvoir générer des IHM plus puissantes.

Bibliographie

Augsbourg(d'), Bruno, 1998, Using Model Checking for the Automatic Validation of User Interfaces Systems. In Design, Specification and Verification of Interactive Systems, Springer

Palanque, Philippe, Bastide, R., 1998, Synergistic modelling of tasks, users and systems using formal specification techniques, Interacting with Computers 9

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

Paternò, F., Frosini, L., Manca, M., 2013, A framework for the development of distributed interactive applications, EICS '13 Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems

Liste des communications

Formal H 2012 Workshop, London, UK, Acceptée, Présentée

IHM 2013, Bordeaux, France, Refusée