A formal language for next generation cockpits user interfaces specification

Vincent Lecrubier
ONERA/DTIM
Toulouse
vincent.lecrubier@onera.fr

Bruno d'Ausbourg
ONERA/DTIM
Toulouse
ausbourg@onera.fr

Yamine Aït-Ameur
ENSEEIHT
Toulouse
yamine@enseeiht.fr

ABSTRACT
User interfaces take an important role in commercial success of past and future aircraft programs. They have a direct influence on flight safety, crew members opinion of the aircraft and the efficiency of aircraft operations. As computing power and complexity of systems increase, user interfaces must become more and more effective in order to allow human operators to deal with this complexity. The goal of this project is to develop a formal language which will enhance the development process,verification and validation of these user interfaces.

Author Keywords
Human Machine Interface; User Interface; Design; Specification; Verification; Validation; Formal Language.

ACM Classification Keywords
H.5.2. User Interfaces: Evaluation/methodology

General Terms
Design; Human Factors; Languages; Reliability; Security; Verification.

ORIGIN AND UNDERLYING PRINCIPLES
Critical embedded UIs (*user interfaces*) conception is a discipline which lies at the limits of two software engineering domains: embedded software development and user interface development. As a consequence, critical embedded user interfaces must comply with a set of conflicting constraints coming from these two different domains.

While embedded software makes robustness a priority, UIs must be flexible and configurable. While embedded systems have limited ressources, UIs must be complete and integrate lots of functionalities. While critical software have strong timing constraints, UIs must be user friendly and let the user interact at his own pace. While critical systems are often real-time, UIs are mostly event-driven.

This set of conflicting constraints is a strong limitation for critical UIs, and makes their development particularly costly and time-consuming.

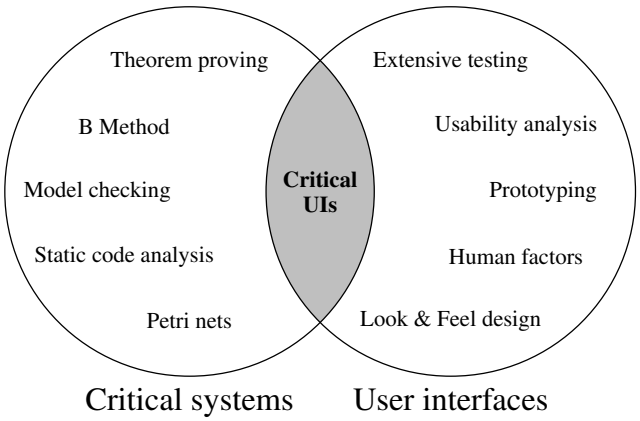

Figure 1. Critical UIs : collision of two clashing domains

The apparition of model-based development has been a big leap ahead for non-critical UI development, allowing for more complex UIs to be designed in a quicker and cheaper way. However, critical UIs did not really profit yet from this new fresh air. They indeed lack development methodologies and tools which would be compatible with safety requirements while allowing quick and efficient development.

A good critical UIs development process should take into account the methods and tools in use for both of these domains (see Fig. 1).

This project tends to make a constructive use of formal methods in critical UIs development. As so, it relies on previous works on this subject. Many works were already performed on formal modelling of software aspects of UIs by suggesting two main approaches.

The first one is based on proof systems where a model of the system is described by sets of variables, sets of operations, sets of events, timing properties and invariants. Operations must preserve invariants and properties. To ensure the correctness of these specifications, proof obligations are generated and must be proved. So, for example, Z and VDM were used to define atomic structures of interactions ([11],[12]) and HOL (High Order Logic Theorem Prover) was used to check user interface specification ([7]). B system was also used for the an incremental specification design of interactive systems ([3], [2]).

The second one is based on the evaluation of logical properties of a system on a state transition system. This technique was used, for example, to verify formally with SMV some

Submitted for review.

properties of interactive systems ([8]). Model checking was used in ([14],[5]) where the user and the system are modelled by object Petri nets (ICOs). In [15], Murϕ is used for modelling an autopilot system and a mental model in order to analyse cockpit interfaces and to detect some potential problems. Model checking was also coupled with static analysis of program codes in [10] or [9] to extract and abstract a formal model of an interactive system and to check various properties on it modelling and analysis of human-automation interaction has also been demonstrated [6].

This project aims at inverting the approach in [10] or [9] by defining a language which would allow to formaly describe model the behaviour of UIs. The model perimeter is the *dialog controller* and *Logical interaction* of the UI Arch Model from [4]. The model will be using abstractions to perform model checking on the high level behaviour of UIs.

MODELLED RELATIONSHIPS

The language goal is to model the UAs (*user applications*) which are applications that comply with the ARINC 661 Specification. ARINC 661 is the industry standard for Cockpit Display System Interfaces to User Systems [1].

This formal language would take into account aspects of both abstract UIs and embedded systems :

- UIs (structure, internal behaviour, human interaction, interface with systems)
- Critical systems (resources footprint, timing constraints, reliability, integrity)

A typical model specified using this language would be a closed system, with a static structure, interacting with two types of abstract external objects (see Fig. 2):

- Display System (e.g. CDS (*Cockpit Display System*))
- Application (e.g. Other aircraft application)

The system would be specified by organizing abstract components in order to generate the desired behaviour. The modelled system would then allow to generate different projections:

- Models to be checked
- ARINC 661 compliant code
- Test cases and scenarios for validation

PROBLEMS ADDRESSED

As of today, in order to pass qualification, UIs must undergo a massive, long test procedure, like many other pieces of embedded software. However, the fact that UIs interact directly with humans makes the conception process much heavier. Indeed, fully automatic generation of UIs still does not yield satisfactory results, and extensive automatic testing is not possible (see Fig. 3).

As a consequence, UIs have to be tested manually during the verification and validation phases. Since these tests happen at the end of the conception loop, errors found during manual tests can become excessively costly to correct, needing to get back in the conception loop.

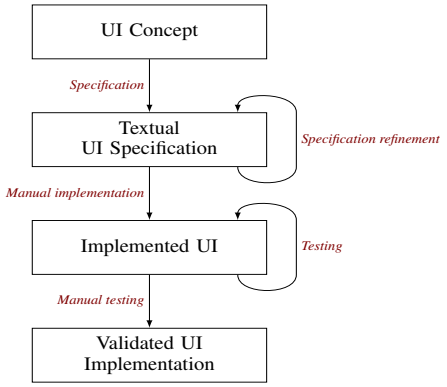

Figure 3. Classical UI conception process

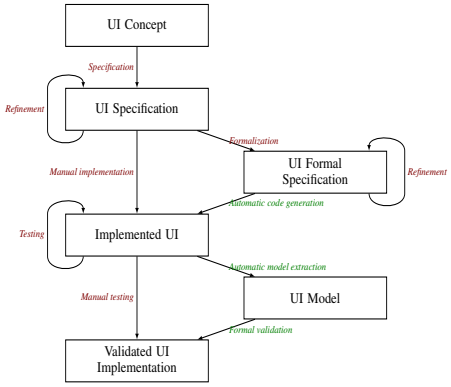

Figure 4. State-of-the-art UI conception process

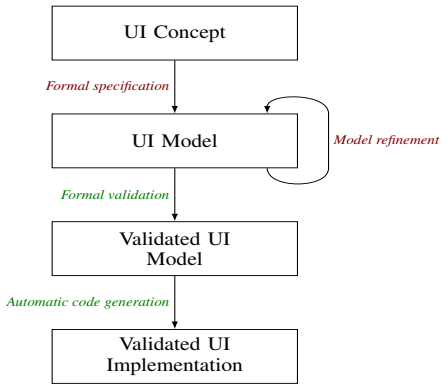

Figure 5. Proposed UI conception process

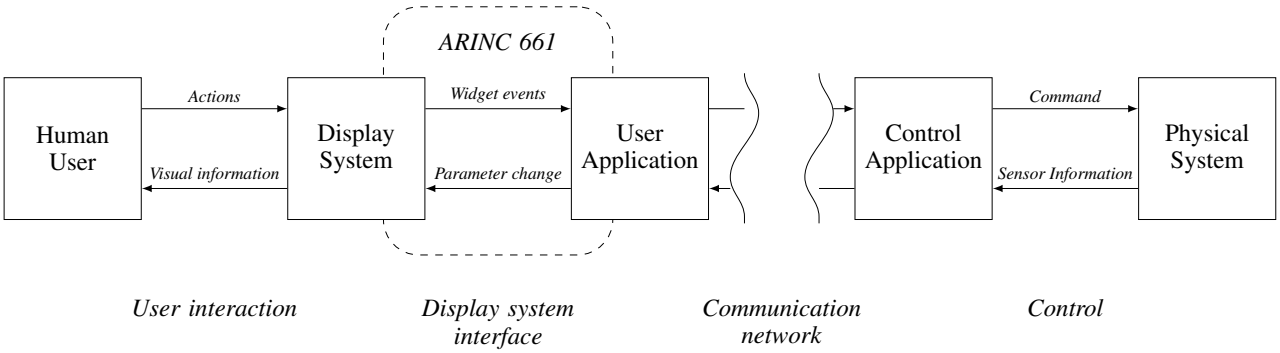

Figure 2. Aircraft UIs architecture. The UA interacts with the CDS one one side, and with other aircraft applications on the other side.

New tools have been developed in order to enhance this heavy process. Some of these tools can be used in order to generate an abstract model of the UI, taking its code as an input, allowing to perform formal validation on the implemented UI [9]. Other methods allow to specify UIs using specific languages in order to validate critical parts [13].

However, a common language allowing an efficient collaboration of these different tools does not exist yet. That is why critical UIs development process can become complex and costly (see Fig 4).

Progress toward a straightforward conception process (see Fig 5) is required to the conception of future UIs. The existence of a formal language taking into account all the aspects of critical UIs (see Fig 1) would greatly help to fulfil this goal.

APPLICATIONS

The application field of this formal language will be formal specification, verification and validation of ARINC 661 compliant UAs.

By enhancing the development process, this approach aims at reducing development costs of critical UIs. The approach could allow to shorten testing phases and to reduce development cycles time.

LIMITATIONS AND DEVELOPMENT OPPORTUNITIES

The language will be limited to embedded critical UIs complying with the ARINC 661 specification. Thus, the following limitations will exist:

- Graphical user interfaces only
- No dynamic instantiation of widgets (UI structure specified at conception time, no modifications at run time)
- No concern about rendering and other low level concerns (Dealt with the CDS, outside of our scope)

In a first approach, we will target the work around modelling the complexity of UI dynamic behaviour, dropping the following concerns:

- Look&Feel (formatting, colouring, screen structure)
- Human factors (usability analysis, ergonomics)

However, the language will be designed to be as expandable as possible, in order to ease future developments.

REFERENCES

1. ARINC. *Specification 661*, supplement 5, draft 1 ed., March 2012.
2. At-Ameur, Y., Brhole, B., Girard, P., Guittet, L., and Jambon, F. Formal verification and validation of interactive systems specifications. from informal specitications to formal validation. In *Conference of Human Error, Safety and Systems Development, HESSD* (Toulouse, 2004).
3. At-Ameur, Y., Girard, P., and Jambon, F. Using the B formal approach for incremental specification design of interactive systems. In *Engineering for Human-Computer Interaction*, S. Chatty and P.Dewan, Eds., vol. 22, Kluwer Academic Publishers (1998), 91–108.
4. Bass, L., Little, R., Pellegrino, R., Reed, S., Seacord, R., Shepard, S., and Szezur, M. The arch model: Seeheim revisited. In *Proceedings of the 1991 User Interface Developers Workshop*, ACM (Seeheim, 1991).
5. Bastide, R., Navarre, D., and Palanque, P. A tool-supported design framework for safety critical interactive systems. *Interacting with Computers 15*, 3 (juin 2003), 309–328.
6. Bolton, M., and Bass, E. Evaluating human-automation interaction using task analytic behavior models, strategic knowledge-based erroneous human behavior generation, and model checking. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* (oct. 2011), 1788 –1794.
7. Bumbulis, P., Alencar, P., Cowan, D., and Lucena, C. Validating Properties of Component-Based Graphical User Interfaces. In *Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS96)*, S. Verlag, Ed. (1996), 347–365.
8. Campos, J. C., and Harrison, M. D. Formally verifying interactive systems: A review. In *Eurographics Workshop on Design, Specification, and Verification of Interactive Systems (DSV-IS97)*, M. D. Harrison and J. C. Torres, Eds., Springer (1997), 109–124.

9. Cortier, A. *Contribution à la validation formelle de systèmes interactifs Java*. PhD thesis, Universite Paul Sabatier - Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-Supaero), 2008.
10. d'Ausbourg, B., Seguin, C., and Roché, P. Assisting the automated validation process of user itnerfaces. In *The 20th International Conference on Software Engineering* (Kyoto, Japan, 1998).
11. Duke, D., and Harrison, M. Abstract Interaction Objects. In *Proceedings of Eurographics conference and computer graphics forum*, vol. 12 (1993), 2536.
12. Duke, D., and Harrison, M. Towards a Theory of Interactors. Tech. Rep. 7040, Amodeus Esprit Basic Research Project, 1993. System Modelling/WP6.
13. Madani, L. *Utilisation de la programmation synchrone pour la spécification et la validation de services interactifs*. PhD thesis, Université Joseph Fourier - Grenoble 1, 2007.
14. Palanque, P. A., Bastide, R., and Sengès, V. Validating interactive system design through the verification of formal task and system models. In *Proceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction*, Chapman & Hall, Ltd. (London, UK, UK, 1996), 189–212.
15. Rushby, J. M. Analyzing cockpit interfaces using formal methods. *Electr. Notes Theor. Comput. Sci. 43* (2001), 1–14.