Cost-efficient design and production of flexible and re-usable near real-time tactical human-machine interfaces

O Grisvard, C Huntzinger, M. Le Berre, V Verbeque

Cost-efficient design and production of flexible and re-usable near real-time tactical human-machine interfaces

O. Grisvard[1,2], C. Huntzinger[1], M. Le Berre[1], V. Verbeque[1]

1: THALES Airborne Systems, 10 Avenue de la 1ère DFL – CS 93801 – 29238 Brest Cedex 3 – France
2: ENST Bretagne, Technopôle Brest-Iroise – CS 83818 – 29238 Brest Cedex 3 – France

Abstract: Making complex systems accessible to human operators supposes to design HMIs that provide the operator with means to manage the complexity in an efficient manner. This is particularly true in the aeronautics domain for tactical HMIs where complexity is present in many dimensions. Current technical requirements, such as being able to display thousands of objects updated on the basis of time intervals inferior to half a second, coupled with economic requirements such as manning and cost reductions, make this issue even more crucial. We present our approach to the design and production of near real-time tactical HMIs, that enables us to devise HMIs that meet such requirements while being flexible enough to be re-used in a wide variety of contexts and produced at a reasonable cost.

Keywords: human-machine interface, near real-time, design, cost-efficiency, re-use.

1. Introduction

Making complex systems accessible to human operators supposes to design human-machine interfaces (HMIs) that provides the operator with means to manage the complexity of the system in an efficient manner. This is particularly true in the aeronautics domain for tactical command and control (C2) HMIs whose purpose is to present the operator with a synthetic and comprehensive view of the situation and where complexity is present in many dimensions: diversity of sensors, diversity of missions, diversity of environments (aircrafts, ships…), diversity of crew configurations. Current technical requirements, such as being able to display thousands of objects updated every half-second while remaining highly fault-tolerant to ensure stability for the operator under stress, coupled with economic requirements such as manning and cost reductions, make this issue even more crucial.

We present here our approach to the design and production of near real-time tactical HMIs, that enables us to devise HMIs that meet such requirements while being flexible enough to be used in a wide variety of contexts and produced at a reasonable cost. This approach associates rapid prototyping, in order to verify the compliance with user requirements and to identify dimensioning parameters, together with an innovative framework for HMI production based on state-of-the-art architectural patterns and HMI technologies.

We demonstrate our approach with examples of HMIs in the domain of C2 systems for maritime patrol and surveillance. Maritime patrol (military) or surveillance (civilian) consists in embarking operators in the cabin of an aircraft in order to survey coastal waters and identify the mobiles at the surface of the sea for various purposes. The corresponding HMI examples are taken from ongoing industrial programs in this domain.

2. Managing complexity

In the domains for which we develop mission systems, complexity is present in many dimensions that all impact the HMI. To begin with, there is an intrinsic functional complexity of nowadays C2 systems. For example, a state-of-the-art maritime patrol system is based on the combination of many sensors: radars (classic radar, SLAR – sideways looking airborne radar), TV/IR camera (infra-red), electro-magnetic detection, IFF (identification friend or foe), UV/IR scanner… Consequently, such a system can propose up to more or less six hundred different functions. The functional complexity of the resulting HMI is of course proportional to that intrinsic complexity.

C2 systems are used for a wide variety of missions, depending on the domain of application, some of them being common to all domains, such as tactical situation elaboration (TSE), the others being specific to each domain. For example, in the domains we work for, the missions include the following:

- Maritime patrol (military): TSE, anti-surface warfare, anti-submarine warfare, joint littoral warfare, targeting for cruise missiles;
- Maritime surveillance (civilian): TSE, exclusive economic zone control, maritime traffic control, fishery control, fight against pollution, contraband, piracy, illegal immigration and drug trafficking, protection of installations (offshore, oil rigs, ports…), maritime search & rescue;

- Electronic warfare (military): TSE, radar detection, communication detection, radar jamming, self-protection, electronic intelligence.

The mission system must be designed as to support the performance of these various missions, the HMI being the central access point for the operator(s) that must perform them.

Additionally, mission systems are embarked on various platforms with heterogeneous working environments, for example aircrafts, helicopters, frigates or submarines. Each platform has its own set of physical constraints that have a direct impact on HMI design, for example:

- Available room: constrains the ergonomics;
- Lighting: constrains the graphics;
- Movement, vibration: constrains the use of input devices;
- …

Finally, each domain also brings specific requirements for crew configurations. The number of operators may greatly vary. It ranges from one operator who performs all the work on board a small maritime surveillance plane, to nine operators on board the biggest maritime patrol aircraft, for which the work is broken down among the team members. In addition, the crew configuration may change during the mission, with operators coming in and out depending on the urgency of the situation. The workload has then to be dynamically re-dispatched and the mission system reconfigured accordingly. Additionally, the system must be highly fault-tolerant as to preserve stability for the operator under stress in critical situations. Being the operators' point of entry into the system, the HMI is directly impacted by that aspect of complexity management.

The current economic constraints impose to reduce as much as possible the number of operators. This implies to solve the difficult equation of enabling an ever smaller number of individuals to cope with an ever increasing system complexity. The solution lies in a design of the HMI that will make the functional complexity affordable to the operator, which is not a trivial issue (see [1] for instance).

Maximising the efficiency of the operator has a direct technical impact as this means to accordingly increase the efficiency of the system while maintaining the standards in terms of reliability. A striking example is the requirements imposed on the HMI for tactical situation management. It is common nowadays to require the processing of thousands of moving graphical objects that are updated on the basis of time intervals inferior to half a second (see Figure 1 for an example of the resulting view). This of course has to be done while preserving the interactivity on these objects, such that there is no perceptible latency for the operator that manipulates them.

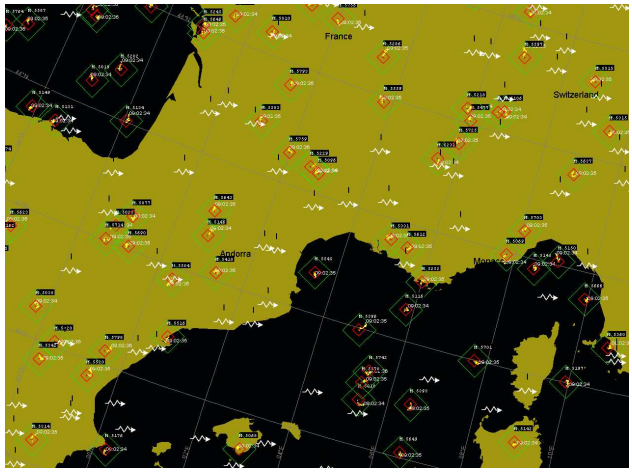

Figure 1: A typical tactical situation

Table 1 below gives examples of time requirements taken from a real case of a tactical HMI for maritime patrol or surveillance. The terms "near real-time" (NRT) comes from such requirements. Fulfilling these requirements imposes to devise new architectural and technical solutions in order to optimise data processing at each stage of the process, from data production by the sensor to visual rendering of the data on the graphical display. This also results in an increased complexity of the HMI, both at the architectural level and in terms of technical solutions. In particular, tactical HMIs have to perform automatic data fusion from various sensors, merging for example video streams with synthetic objects. Being compliant with NRT requirements in such a context imposes to devise specific architectural patterns.

Action	Response time definition	Time (sec)
Window move	From command to resulting display	0.5
Key press	From key press to visual feedback	0.1
Key print	From key press to character display	0.2
Page scroll	From end of request to beginning of text scroll	0.5
X/Y entry	From selection of field to visual verification	0.2
Pointing	From pointing input of to visual feedback	0.2
Sketching	From pointing to line display	0.2
Local update	Time for updating the screen image with data from the local cache	0.5

Table 1: HMI response time requirements

All the dimensions of complexity of mission systems, both intrinsic and induced by their context of use, are reflected in the corresponding design and production process. A great number of actors with quite different

profiles play a part in that process: system architects, system engineers, technical experts, software architects, software engineers, software developers, etc. Even the customer is now involved, both through end-users and domain experts. There also an increasing appeal to sub-contracting in order to lower the production costs. Indeed, due to market globalisation, cost reduction becomes mandatory in order to remain competitive. This is another difficult equation to solve, producing systems ever increasing in size and complexity while avoiding the explosion of costs if not actually reducing them.

In the following sections we present the approach that enables us to tackle the issues outlined above that rise from the problematic of near real-time HMI production for complex mission systems. Our approach is based on a precise separation of concerns through a step by step process from the HMI definition to its integration in the system, together with a multi-iterations process, iterating first on definition and prototyping, then on modelling and architecting, and finally on development and test (see Figure 2).

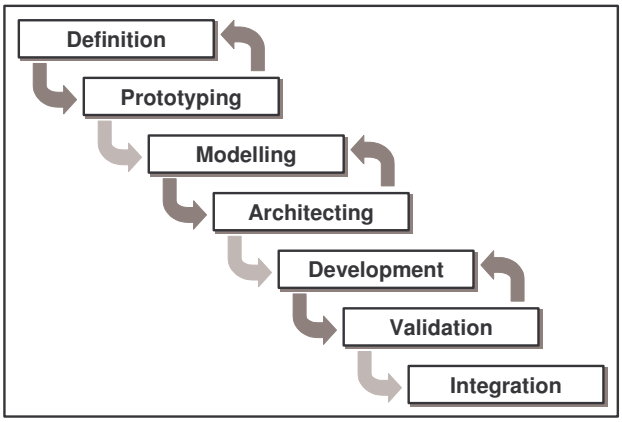

Figure 2: The HMI design and production process

The three following sections detail each iteration step of the process.

3. Rapid prototyping

As mentioned above, one of the main issues of the design of complex systems is to be able to design HMIs that make the complexity of the system affordable to the users. Traditional approaches to system design focus on the system's functions, the HMI being treated as a side task, not being particularly challenging. As a matter of fact, it is still nowadays often heard that HMI design is an easy task and that HMI production should cost nothing. This is a major mistake when addressing tactical command and control systems that are by definition HMI-centric. This has often led to bad HMI design and the rejection of the HMI by the end-users, with consequently important additional costs in order to reach customer satisfaction. This is even more crucial as the complexity of these systems explodes. If one does not allocate sufficient time an effort, the risk is very high to design unusable HMIs. As such, as costs must be kept reasonable, HMI design has to be re-thought completely [2].

A solution that has proven efficient and affordable in terms of costs is to rely on rapid prototyping to iterate with the end-users during HMI definition in order to ensure the usability of the resulting HMI (see [3]). The additional initial costs induced by this approach are largely counterbalanced by the costs savings in the production process, as the HMI design is agreed once and for all at the end of the first iteration step. The prototype can then be used as a contract between the customer and the supplier in order to guarantee the end-user that the final HMI will conform to that agreement.

To support the definition step of our process, we have designed a solution for rapid HMI prototyping dedicated to tactical HMIs for C2 systems called SMOCK (for "Smalltalk MOCK-up"). At the prototyping step, the focus has to be put on easing the design and limiting the time required to update that design. Therefore, we have based our solution on the Smalltalk language (www.smalltalk.org) due to its essential property to integrate the model and the code, that has proven particularly suited for rapid code change. Figure 3 shows an example of a tactical HMI prototype produced with SMOCK.

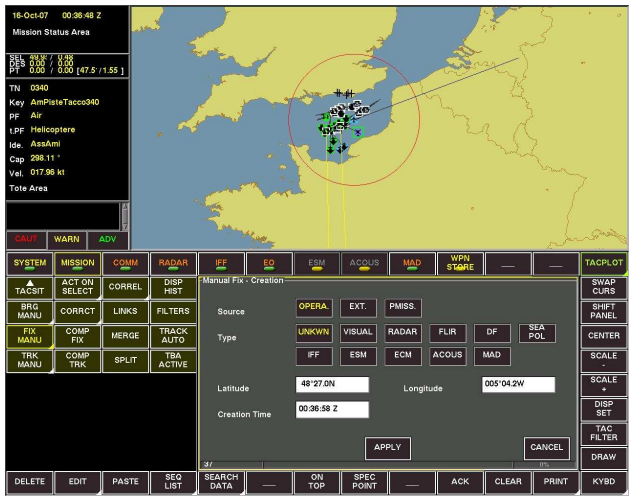

Figure 3: A tactical HMI prototype

SMOCK enables HMI designers:

- To simulate the behaviour of the HMI and the interactions between the operator and the system, validating not only the graphical design but also the ergonomics of the HMI;

- To mock-up several complete operator positions and to evaluate the whole in a multi-operator environment (see Figure 4 below), easily testing different crew configurations;
- To share this vision with all the actors, including the customer, and proceed to light-cost iterations in order to reach an agreement on the HMI definition;
- To simulate early dimensioning scenarios, in particular by integrating the calculation of a cost-model in order to evaluate the financial impact of a choice in terms of ergonomics;
- To pre-model the HMI architecture, outlining the basic building blocks or components and the interfaces between these components as well as identifying at an early stage in the process the important issues regarding efficiency, resource consumption, data-flows, etc.

Figure 4: A multi-user prototype

SMOCK maximises the productivity enabling short iterations with the customer on the HMI definition. As an example, it takes from one minute to change a button up to one day and a half to update about ten panels. With SMOCK, prototyping becomes a light-weight process, as the tool can be installed rapidly on a laptop and frequently and easily updated. Finally, it benefits from additional properties of Smalltalk, for example openness to other technologies, rapid adaptation to any type of platform and compatibility with legacy software.

SMOCK has been used successfully in real-sized programs for maritime patrol and surveillance systems. The images above are taken from one of these programs. One important benefit of SMOCK in terms of cost reduction is that the prototypes developed for a given program can be straightforwardly re-used for another program in the same domain as they do not have to cope with hardware and technical requirements and can be easily and rapidly adapted.

4. A generic HMI framework

Following prototyping, we find the central steps of the process that consist in modelling the HMI and defining its software architecture. These steps are crucial as they condition the code production process by identifying the building blocks and the principles on which the code decomposition and assembly will be based. They are initiated during the architecture pre-modelling phase of the prototyping step.

Our approach to modelling and architecture falls within the scope of the model-driven engineering (MDE) and model-driven architecture (MDA) methodologies, as established standards for the development of complex software systems (see [4]). The MDE approach places the model as the centre of the software production process. The model contains the necessary information to ensure the consistency of the source code, the testing of that code, and the associated documentation (see Figure 5). Additionally, following a MDA approach, we also include in the model the architectural patterns that will guarantee both the efficiency of the HMI and the manageability of the HMI code.

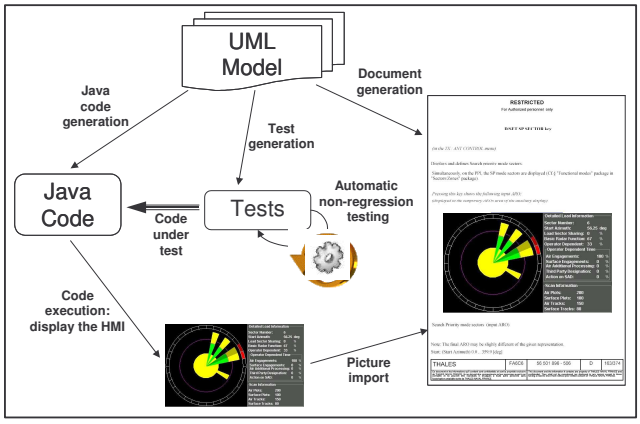

Figure 5: The MDE process for HMI

Our approach is implemented as a specific layer dedicated to tactical HMI production (GFL for Graphical Framework Layer) on top of a generic framework for HMI production called JAGUAR (JAva Graphic Unified ARchitecture) developed conjointly with another division of THALES now part of DCNS. JAGUAR/GFL is implemented with up-to-date Java technology (java.sun.com). The models are expressed using the UML standard (www.uml.org),

and code generation from the model relies on a direct mapping of the UML class diagram into a Java class hierarchy.

JAGUAR/GFL relies on a component-based approach to HMI architecting. In JAGUAR, components are realised on the basis of JACOMO (JAguar COmponent MOdel), which is a lightweight component model derived from the CCM (CORBA Component Model, see www.omg.org/technology/documents/formal/components.htm) specification. In JACOMO, components are characterised by the following elements (see Figure 6 below):

- The services they provide (facets) and use (receptacles);
- The events they produce (event sources) and consume (event sinks);
- A clear separation between their interface and their implementation;
- A customisable behaviour using parameters (attributes).

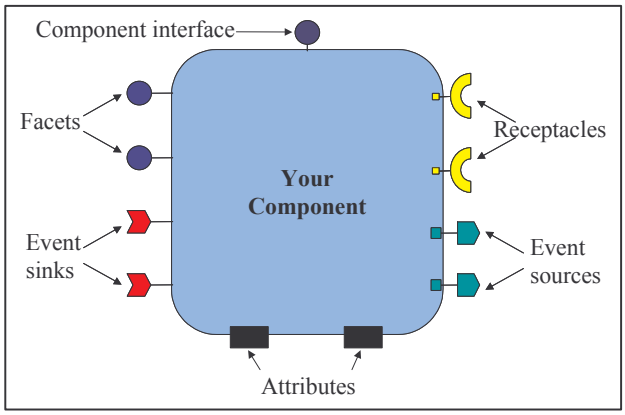

Figure 6: The component model

Such components facilitate the decomposition of the software and enable:

- Explicit contracts as services provided and services required, event and data flows;
- Easy abstraction from technical details (for instance, functional components vs. technical components) and from specific implementations.

A direct consequence in terms of HMI production management is that components provide direct support to:

- The HMI code production organisation with facilitated implementation task distribution and explicit contracts, especially useful when involving subcontracting;
- The HMI software validation process with black-box unitary testing at the level of elementary components;
- The HMI software management with an independence from COTS components (for example the cartographic component) and an optimised re-usability of software components.

Building on the component-based approach, GFL implements a new architectural pattern dedicated to the highly demanding processing of tactical graphical HMIs. This pattern is an enhanced version of the traditional model-view-controller (MVC) pattern for HMIs (see Figure 7).

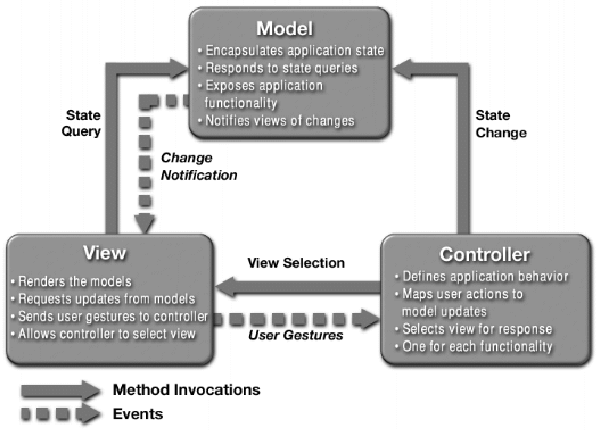

Figure 7: The MVC pattern

This pattern supports:

- Independence between the various HMI processes (operator inputs, displays, data updates);
- Optimal management of the various HMI tasks;
- Isolation of tasks that are costly in time or in resources.

The benefit for the operator is the preservation of the interactivity even in the case of a highly loaded situation such as the one of Figure 8 (page 6).

Relying on that specific architectural solution, we guarantee the following capacities to the operator:

- A maximum reactivity on interactions;
- No global slowdown or freezing of the HMI;
- Fault tolerance and robustness to breakdowns.

The quality, the efficiency and the robustness of the HMI is thus maximised, which enables us to ensure the processing of about fifteen hundred tactical objects during six hours of mission consuming only 5% to 8% of the available CPU time. This has been measured on an up-to-date dual-core processor together with a standard video card.

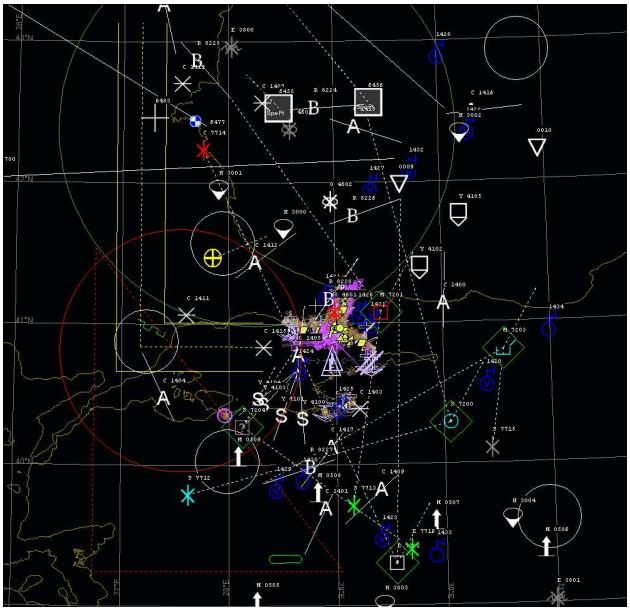

Figure 8: The standard load of a tactical view

Using the JAGUAR/GFL framework, we have developed a library of generic tactical HMI components that can be re-used across programs even in these concern different application domains. Indeed, we have already experimented the effective re-use of components developed for a maritime patrol system in the context of an electronic warfare system with HMIs that have different look and feel but share similar efficiency and interactivity requirements. Capitalisation with JAGUAR/GFL has been evaluated as such:

- 35% of re-use intrinsic to the generic HMI production framework;
- 35% of additional re-use of graphical HMI components;
- Only 30% of the code remaining to be produced for each specific HMI.

These figures clearly demonstrate a high capacity to reduce the costs of HMI production after a first initial development.

5. Performing validation

As mentioned above, the component-based approach we follow allows for unitary testing of individual components as black-boxes. System-wide testing is also facilitated by that approach as individual components can be tested in the system context using simulation of the other components or even simulation of the software infrastructure. Test productivity is thus increased by the capacity to perform parallel and incremental testing of the HMI during its implementation, not having to wait for the whole HMI to be developed before the validation phase can start.

Furthermore, our approach to HMI validation targets the automation of tests, once again in order to reduce the costs. We use the prototype model in order to identify the characteristics of the HMI that must be tested. Test scenarios can be generated from experimentations with the prototype using logs of the operator's actions. These scenarios can then be used by our validation environment to go through all the necessary HMI testing. This environment is currently being implemented in Smalltalk, like our prototyping framework. This will enable us to effectively couple the two in order to implement the test automation process.

6. Conclusion

We have presented here our approach and process for NRT tactical HMI design and production. Our solution integrates a rapid prototyping tool, a generic tactical HMI modelling and production framework and a validation environment. All the architectural and technical choices we have made have been driven by the need to manage the complexity of such HMIs, while preserving the usability for the operator and limiting the production costs.

We have demonstrated how our approach ensures that end-user requirements are appropriately taken into account, guarantees the efficiency of the resulting HMI and maximises the re-use from one development to the other.

Our current activity is focussed on the automation of tests as described above. The main foreseen perspective to our work is to study a stronger coupling between the prototyping phase and the modelling and architecting steps. The idea consists in re-using the model and the first architecture produced during the prototyping phase as a basis for the production of the final model and architecture of the HMI. This should enable us to save time and efforts for the production of the final HMI model and functional architecture.

7. References

[1] Boy G. & Bradshaw J.: *"Perceived complexity versus internal complexity. Did we take into account expertise, reliability and cognitive stability?"*, Second Symposium on Resilience Engineering, Juan-Les-Pins, France, 2006.

[2] Sedogbo C., Bisson P., Grisvard O., Poibeau T.: *"Human-System Interaction Container Paradigm"*, Tenth International Conference on Human-Computer Interaction (HCI International), Heraklion, Crete, Greece, 2003.

[3] Hardtke F.: *"Rapid prototyping for user-friendly and useful human-machine interfaces"*, SIAA Annual

Simulation Technology and Training Conference (SimTecT), Canberra, Australia, 2001.

[4] France R. & Rumpe B.: *"Model-driven development of complex software: a research roadmap"*, International Conference on Software Engineering: Future of Software Engineering, Minneapolis, USA, 2007.

8. Glossary

C2: Command and Control
CCM: CORBA Component Model
COTS: Commercial Of-The-Shelf
CPU: Central Processing Unit
GFL: Graphical Framework Layer
HMI: Human-Machine Interface
IFF: Identification friend or foe
MDA: Model-Driven Architecture
MDE: Model-Driven Engineering
MVC: Model View Controller
NRT: Near Real Time
SLAR: Sideways Looking Airborne Radar
TSE: Tactical Situation Elaboration
TV/IR: Tele-Vision/Infra-Red
UML: Unified Modelling Language
UV/IR: Ultra-Violet/Infra-Red