Tool Support for Interactive Prototyping of Safety–Critical Interactive Applications

Rémi Bastide
Université Paul Sabatier, France

David Navarre
Université Paul Sabatier, France

Philippe Palanque
Université Paul Sabatier, France

INTRODUCTION

The complete specification of interactive applications is now increasingly considered a requirement in the field of software for safety-critical systems due to their use as the main control interface for such systems. The reason for putting effort in the use and the deployment of formal description techniques lies in the fact that they are the only means for both modeling in a precise and unambiguous way all the components of an interactive application (presentation, dialogue, and functional core; Pfaff, 1985) and proposing techniques for reasoning about (and also verifying) the models (Palanque & Bastide, 1995).

Formal description techniques are usually applied to early phases in the development process (requirements analysis and elicitation) and clearly show their limits when it comes to evaluation (testing).

When the emphasis is on validation, iterative design processes (Hix & Hartson, 1993) are generally put forward with the support of prototyping as a critical tool (Rettig, 1994). However, if used in a nonstructured way and without links to the classical phases of the development process, results produced using such iterative processes are usually weak in terms of reliability. They can also be unacceptable when interfaces for safety-critical applications are concerned.

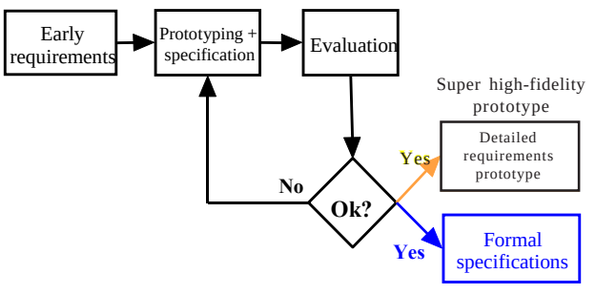
If we consider interfaces such as the ones developed in the field of air traffic control (ATC), a new characteristic appears, which is the dynamics of interaction objects in terms of existence, reactivity, and interrelations (Jacob, 1999). In opposition to WIMP (windows, icons, menus, and pointing) interfaces, in which the interaction space is predetermined, these interfaces may include new interactors (for instance, graphical representations of planes) at any time during the use of the application (Beaudouin-Lafon, 2000). Even though this kind of problem is easily mastered by programming languages, it is hard to tackle in terms of modeling. This is why classical description techniques must be improved in order to be able to describe in a complete way highly interactive applications.

BACKGROUND

Several approaches propose solutions for the reconciliation of the specification and the validation phases in the field of interactive applications, but these solutions are often incomplete according to three different viewpoints.

- **Interaction Style Viewpoint:** Post-WIMP user interfaces are not yet widely developed. For this reason, most of the approaches (see, for instance, Hussey & Carrington, 1999) only deal with WIMP interfaces, that is, static interfaces for which the set and the number of interactors is known beforehand. The behaviour and the role of these interactors are standardised (typically windows and buttons belong to this category).
- **Development Phase Viewpoint:** We often find disparate solutions that do not integrate the various phases in a consistent manner (Märtin,

Figure 1. Iterative development process with PetShop



1999). So, most often, several gaps remain to be bridged manually by the teams involved in the development process.

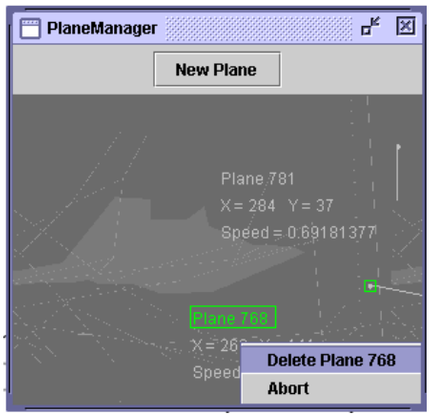
- **Reliability of Results Viewpoint:** Several integrated approaches have been proposed for WIMP-interactive applications. Among them, we find TRIDENT (Bodart, Hennebert, Leheureux, & Vanderdonckt, 1993), which is the more successful one as it handles both data and dialogue description and as it also incorporates ergonomic evaluation by means of embedded ergonomic rules. However, specification techniques used in the project have not been provided with analysis techniques for verifying models and the consistency between models.

PROTOTYPING CAN BE FORMAL, TOO

The PetShop (Petri Nets Workshop) CASE (computer-aided software engineering) tool promotes an iterative development process articulated around the use of a formal description technique of the dialogue of the interactive application.

This formal description technique (based on the petri nets) was developed at LIIHS in the early '90s (Bastide & Palanque, 1990) and has been refined since then (Bastide & Palanque, 1999). The use of this kind of modeling technique provides extended benefits with respect to those less formal. Indeed, analysis tools, exploiting the mathematical background of formalism, allow the validation of the application before its implementation.

Figure 2. A menu opened on the radar for a selected plane



A SAFETY-CRITICAL CASE STUDY

The example presented in this article is extracted from a complex application studied in the context of the European project Mefisto (http://giove.cnuce.cnr.it/mefisto.html).

This project is dedicated to formal description techniques and focuses on the field of air traffic control. This example comes from an en route air traffic control application focusing on the impact of data-link technologies in the ATC field. Using such applications, air traffic controllers can direct pilots in a sector (a decomposition of the airspace).

The radar image is shown in Figure 2. On the radar image, each plane is represented by a graphical element providing air traffic controllers with useful information for handling air traffic in a sector.

Figure 3 presents the general architecture of PetShop. The rectangles represent the functional modules of PetShop. The document-like shapes represent the models produced and used by the modules.

PetShop features an object petri-net editor that allows for the editing and executing of the ObCSs (object control structures) of the classes. At run time, the designer can both interact with the specification and the actual application. These are presented in two different windows overlapping in Figure 4. The window PlaneManager corresponds to the execution of the window with the object petri net underneath.