Towards a Formal, Model-Based Framework for Control Systems Interaction Prototyping

Matteo Risoldi[1] and Vasco Amaral[2]

[1] Université de Genève
24, Rue Général-Dufour
CH 1211 Geneva 4
matteo.risoldi@cui.unige.ch
[2] Universidade Nova de Lisboa
Quinta da Torre
P 2829-516 Caparica
vasco.amaral@di.fct.unl.pt

Abstract. This paper provides an overview of a starting project called BATIC^{3}S (Building Adaptive Three-dimensional Interfaces for Critical Complex Control Systems). This project aims to bring a more viable approach in the fields of Graphical User Interfaces (GUI), software modeling and verification, automatic code generation, and adaptivity. The goal is to build a comprehensive methodology for semi-automated, formal model-based generation of effective, reliable and adaptive 3D GUIs for diagnosing control systems. This can be used to assist in GUI development for very complex systems, like industrial systems, high energy physics experiments and similar.

1 Introduction

Developing Graphical User Interfaces (GUIs) for complex control systems is costly, difficult and error prone. Large hardware systems as you can find in industries, physics research centers or transportation (airplanes) have a high complexity coming from the number of components, their hierarchical interaction and the large number of parameters to be monitored at once. This poses challenges in particular concerning the correctness of the control, the navigation issues and the paradigm for human interaction. Therefore, it is of first importance to systematize the specification and modeling of systems, to find new ways of interaction and to minimize the development cost of GUI production in order to be able to test interfaces from a usability point of view.

In this paper we will present a work in progress, stating our views on how various software engineering techniques and aspects can be integrated and coordinated in a methodology to standardize and assist the development of such interfaces. We will speak through examples about what technologies we are using for it. Finally we will briefly illustrate two case studies that are currently guiding our work. A survey on related work, with analysis and a comparative study of previous approaches in the field of user interface prototyping for control systems, has been done in [4].

N. Guelfi and D. Buchs (Eds.): RISE 2006, LNCS 4401, pp. 144–159, 2007.

2 The General Approach

2.1 Definition of a Control System

Control systems (CS) can be defined as mechanisms that provide output variables of a system by manipulating the inputs (from sensors). The field of CS is wide, and ranges from the very simple to the very complex. To avoid a lack of applicability, we had to reduce the field to systems with some specific features. The definition of a CS we use (and a very widely accepted one) is a system which has a hierarchical organization, in which every elementary object can be grouped with others, and composite objects (groups) can be grouped as well, forming a hierarchical tree in which the root represents the whole system and the leaves are the actual devices that form it. Typically this grouping will reflect a physical container-contained composition (but it could follow other relations as well). Elementary and composite objects can receive commands and communicate states and alarms. Figure 1 shows a partial example of such a system, a simple drink vending machine (DVM). We will use it also in following examples.

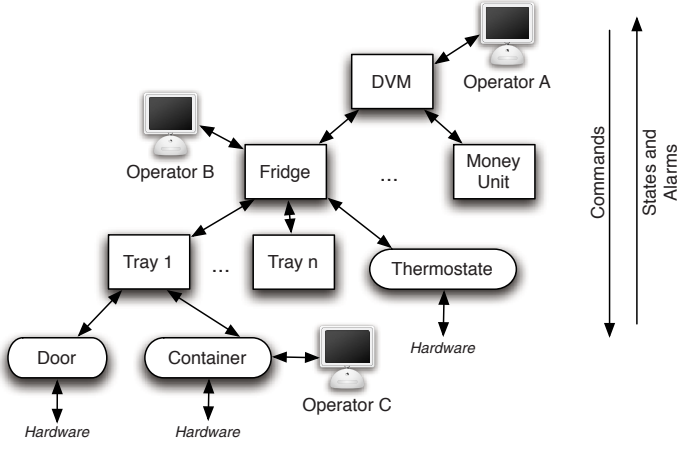

Fig. 1. An example of hierarchical control system: drink vending machine

We can see basic, low level components: the *Door*, the *Container* and the *Thermostate*. These get input directly from the hardware (through sensors, for example), and have simple states which depend on this input (e.g. the Thermostate might have states *TempOk, TempWarning* and *TempError*, depending on a temperature sensor). These basic components are grouped in various ways - Door and Container are grouped in a *Tray*, which is a composite object; several trays and a Thermostate are grouped in a *Fridge*. The latter, and another composite *Money Unit*, are in turn grouped into the top object, *DVM*. We see