Human-system Interaction Container Paradigm

Célestin Sedogbo & Human Interaction Technologies Lab

THALES Research & Technology France
Domaine de Corbeville – 91404 Orsay Cedex – France
[firstname.lastname]@thalesgroup.com

Abstract

In this paper we present the concept of Human Interaction Container (HIC) which introduces an important shift in the field of Human-Computer Interaction, moving from an application-centric to user-centric perspective, through the adoption of a service-oriented view of application and user interface capabilities. The HIC has been designed as to propose an interaction infrastructure offering the necessary decoupling between application, interaction and presentation logics in order to enable intelligent adaptive interaction and easy integration of new interaction modalities and appliances. The HIC approach is in the process of implementation and validation on several THALES business cases such as Collaborative Decision Making for Air Traffic Management.

1 Introduction

Due to the increasing diversity and complexity of software systems and means, the scope of Human-Computer Interaction (HCI) extends far beyond the simple issue of providing human beings with means to use a system. First, HCI must also enable the users to access the system anywhere, anytime and anyhow, that is, more generally speaking, in any context of use. Second, it must provide the users with support and assistance in order for them to perform their task or carry out their mission.

Nevertheless, most existing systems are not directed towards the satisfaction of the users' needs, but rather merely designed as to propose an interface to a set of application functions. Moreover, these functions are often used through the prismatic view of the Graphical User Interface (GUI) and most GUIs do not take into account some key contextual information such as the users' task and behavior and do not exploit all the possibilities offered by the appliances at hands. Therefore, the usability and utility of the User Interface (UI), although being a key acceptance factor for nowadays systems, is often neglected to the detriment of the usability of the whole system.

One of the main stumbling blocks to the design of user-centered systems is the difficulty to clearly identify the interaction logic, that heavily depends on contextual information at several levels (application domain, user tasks, user profiles and preferences, hardware environment and interaction history), as opposed to the business application logic, which should be independent of any context of use, and the presentation logic, which is necessarily specific to a given UI on a given terminal. A consequence of this lack of proper separation between application, interaction and presentation services is that system designers and developers have to face important costs in terms of system deployment, upgrade and maintenance. Indeed, any evolution of the users' interaction needs, for instance when the physical or logical context of use changes, impacts both the applications and their interfaces. We argue that changing this situation in order to open the way to the design of highly interactive systems requires a move from an application-centric perspective, which is the characteristic of most existing systems, to a user-centric perspective.

Moreover, as today interaction means become more and more various and sophisticated, interaction demands the integration of heterogeneous modalities, such as voice, gesture, graphics and animation, as well as appliances, such as classical laptop and desktop workstations, mobile phones, Personal Digital Assistants (PDA), PC tablets, etc. Therefore, HCI systems must undergo an important mutation, moving from a one-to-one to a many-to-many scheme in terms of application to UI pairing, and become able to dynamically adapt or even create the interface according to the users' role and environment, without changing the application.

2 The Human Interaction Container paradigm

A solution for providing users with wider access to existing systems and for enhancing user-friendliness of existing interaction means is to design intelligent interaction systems that dynamically adapt to the interaction environment and react appropriately in various contexts of use, without implying any modification to the core application. In order to achieve this, we propose a new HCI architecture, offering the necessary decoupling between application, interaction and presentation logics in order to implement adaptive interaction. This architecture is based on the key concept of Human Interaction Container (HIC).

The HIC aims at encapsulating all software components dedicated to user-system interaction management into a context-aware and context-sensitive container enacting as a mediator between the application services and the presentation services. As such, this container is designed so to ease the logic separation, between application, interaction and presentation, and handle all the interaction processes enabling an application and its various user interfaces to communicate with each other. It offers application-independent and interface-independent interaction services which support intelligent adaptive interaction. These generic interaction services include dialogue processing, task and activity planning, user adaptation, multi-modality input and output management, multimedia presentation generation and terminal adaptation.

The HIC also aims at providing application and UI designers with an open framework, independent from platforms, networks and appliances, and enabling them to design at compile-time interaction facilities that meet the operator and task requirements and use them at runtime without disrupting the realism of interaction and the user performance. As such, the interaction container can be seen as an attempt to exploit middleware facilities in order to satisfy the interaction demand at the application level. Therefore, from a middleware perspective, the HIC constitutes a set of additional layers, the *interaction layers*, on top of an existing system middleware, this assembly being called an *interaction middleware*. For multi-tier architectures, the introduction of the HIC implies a migration from three tier architectures to four tier architectures, with the HIC as a new tier, inserted between the application server tier and the client tier.

3 Architectures for human-computer interaction

Agent-based frameworks such as the Open Agent Architecture (Cheyer & Martin, 2001) or Ivy (Chatty, 2002) may constitute a first step towards the implementation of the interaction infrastructure we advocate for, as they can be used in order to encapsulate interaction components as agents into an interaction platform independent of either application and interface. Some interaction platforms like the iROS software/middleware system (Ponnekanti, Johanson, Kiciman

& Fox 2003) prefigure the shift towards the new kind of interaction architecture we propose. iROS uses an *event heap* (Johanson & Fox, 2002) as a coordination infrastructure for interactive workspaces. Based on a *tuplespace* model, it offers a satisfactory infrastructure for coordinating and assembling distributed components, even if it lacks several functionalities in terms of event management, especially regarding event life cycle and timestamp management. Moreover, the ICrafter system, (Ponnekanti, Lee, Fox, Hanrahan & Winograd, 2001) designed as a service framework for ubiquitous computing environments, is even closer to our proposal as it is designed as to let users interact with workspace services using a variety of modalities and appliances. The ICrafter architecture is built upon the event-based communication system of iROS and a context memory component used for storing workspace context information.

4 The Human Interaction Container architecture

The architecture we have designed for the HIC results from our background acquired in the field of HCI. It can be seen as a generalization of the approach we pursued in the past when addressing the problem of (semi-)automatic production of speech-based interfaces. This approach consisted in setting-up an agent-based HCI infrastructure where specialized agents devoted to interaction management were communicating through a supervisor. The implementation was done using OAA (cf. section 3). As can be seen on Figure 1, the HIC internal flow processing architecture is also based on a community of agents dedicated to the processing of interaction and implementing the various interaction services, as well as on a resource management facility which ensures real-time access to interaction resources such as activity state, dialogue history or UI contexts. This architecture is supported by the several other layers of the whole interaction middleware which offer the various services required in order to implement high-level interaction processes.

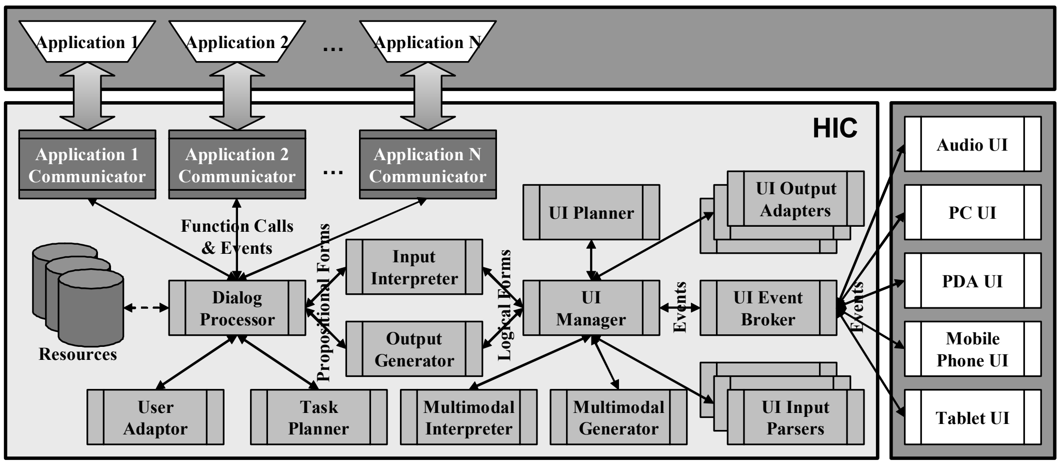

Figure 1: HIC internal flow processing architecture

5 The concept of Interaction Middleware

As stated above (cf. section 2), the HIC can be seen as an interaction middleware whose purpose is to ease the development of HCI systems where people have to interact and possibly collaborate through a heterogeneous set of devices ranging from mobile phones and PDAs to laptop and desktop PCs. The interaction middleware is organized as a set of additional layers on top of a classical system middleware. Such a middleware provides some core services such as object life cycle management, time management and persistence[1]. In our approach, these cores services are completed by two additional layers, one devoted to *technical services* and the other to *interaction services*. As shown on Figure 2, technical services support both the application container and the interaction container while interaction services provide the basis for the implementation of the HIC.

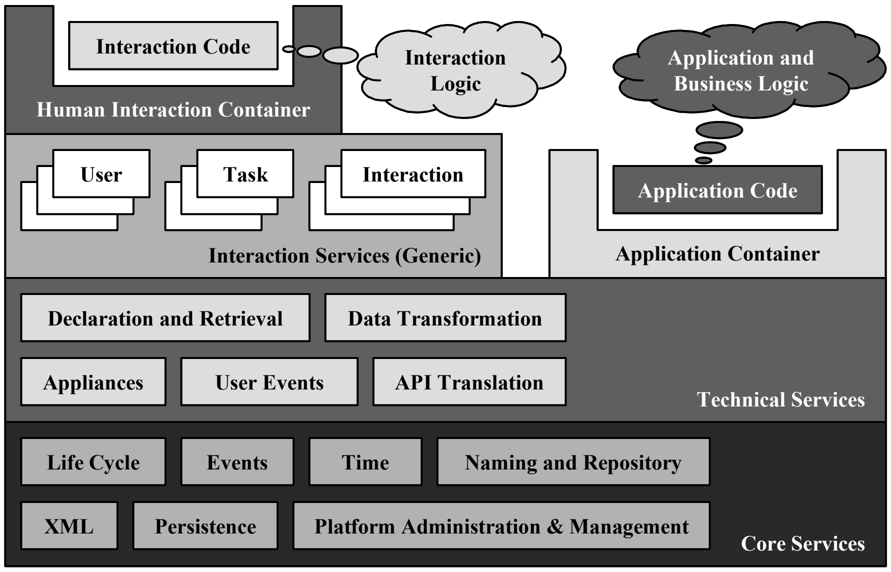

Figure 2: Architecture of an interaction middleware

Technical services are not directly implementing interaction management. They are primary specified and developed in order to be called and used by the upper layer as tools for interaction service implementation. The *declaration and retrieval service* is a facility for the management of the other services (addition, retrieval, removal, etc.). The *API translation service* translates an application API into a set of application services accessible through the interaction middleware in order to ensure compatibility with existing applications. The *data transformation service* offers tools for converting some data from a given format into another. The *user events service* manages user events at different levels of granularity, ranging from elementary events such as mouse moves or key strokes to complex ones reflecting the semantics of user actions such as selection in a list or connection to a device. Finally, the *appliances service* stores the characteristics of the various appliances which are used by some services of the upper layer in order for them to account for appliance variation and react accordingly.

Interaction services are the basis for the implementation of the intelligent interaction capabilities shown on Figure 1 above. These services are concerned with the management of the user task descriptions, the acquisition and update of the user activity state descriptions at runtime, the management of user class and user specific profiles and preferences as well as the management of the interaction (or *dialogue*) state and history.

[1] As core services are common to most existing system middlewares, we will not describe them in details.

6 Implementation and applications

The HIC concept is under a process of implementation by way of an integrated approach that mixes technological development together with operational application. Our first implementation of the HIC technology relies on the iROS middleware (cf. section 3) as a good candidate to experiment our layered and service oriented approach to interaction middlewares. We thus use the event heap to set-up a core service layer that can be seen of as a message-oriented system middleware. A few of the technical services of the second layer are basically the ones proposed by ICrafter but we develop most of these services ourselves, especially in order to support service classification and service hierarchy management. This work is now under completion. Finally, the implementation of the interaction services of the third level as well as the interaction processes of the HIC core will be done on the basis of a model-based approach which enables derivation of interaction model from application model and user task model at-compile-time and management of business interaction patterns and rules at runtime.

While the HIC may address a broader range of applications, its implementation is mainly driven by a set of THALES business cases in which it is meant to support tasks such as Control and Command, Collaborative Decision-Making or Team Situation Awareness, in domains such as Air Traffic Management or Naval Combat Management. It must be noticed that the deployment of the resulting technology, while being strongly dependent on the demand in terms of interaction services, also leads us to design specific architectural patterns per domain and business.

7 Conclusion

Through the adoption of a user-centric approach to interaction management, we argue that the HIC prefigures the future of HCI systems where people will have to interact and collaborate through a heterogeneous set of modalities and appliances. In terms of software development and management, the HIC as an interaction middleware offers the HCI designers an open framework enabling them to design an interaction system that meets business requirements such as enhanced user support and easy integration of new devices, without implying important changes to the application, which is a major requirement of THALES system designers and developers.

References

Chatty, S. (2002). The Ivy software bus – a white paper. CENA Technical Note NT02-816, from http://www.tls.cena.fr/products/ivy/

Cheyer, A., & Martin, D. (2001). The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4 (1), 143-148.

Johanson, B., & Fox, A. (2002). The Event Heap: A Coordination Infrastructure for Interactive Workspaces. *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002).*

Ponnekanti, S. R., Lee, B., Fox, A., Hanrahan, P., & Winograd, T. (2001). ICrafter: A Service Framework for Ubiquitous Computing Environments, *Proceedings of the Ubiquitous Computing Conference (UBICOMP 2001).*

Ponnekanti, S. R., Johanson, B., Kiciman, E., & Fox A. (2003). Portability, Extensibility and Robustness in iROS. *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (Percom 2003).*