Adaptive User Interfaces through Dynamic Design Automation

Robin R. Penner
University of Minnesota
College of Mechanical Engineering
111 Church Street SE
Minneapolis, MN 55455
rpenner@me.umn.edu

Erik S. Steinmetz
University of Minnesota
Dept. of Computer Science and Engineering
200 Union Street SE
Minneapolis, MN 55455
steinmet@cs.umn.edu

Christopher L. Johnson
Honeywell Technology Center
Human Centered Systems
3660 Technology Drive
Minneapolis, MN 55418
chris.l.johnson@honeywell.com

Abstract

The inherent difficulty in supporting human usability in large control systems—such as building environmental and security systems—derives from the large diversity of components and users within each domain. Each system is different, with different types and organizations of devices; each user is different, and takes different roles; each task a user performs varies with the situation. As a result, applying traditional methods of interface design to these systems is insufficient. Designers end up handcrafting each diagram required by each type of user, the effort needed to add new functionality quickly bloats, and users end up juggling multiple disparate applications. We have begun to deploy a tool called DIG (Dynamic Interaction Generation) that addresses this difficulty. DIG uses models of domain, task, and presentation knowledge to automatically design and present interfaces specialized to a user's current role and task, the current situation, and the capabilities of the current display hardware. In this demonstration, DIG will convert a real-life building management configuration into a dynamic interface that building managers can operate using either a standard PC or a Palm Pilot.

Introduction

Designing a user interface is difficult, time-consuming, and expensive. Designing a *good* user interface is even harder. Even though industry support of the practice of UI design has improved in recent years—although still often inadequate—complex, distributed systems pose a particular usability problem. Because of the variability and expandability of these systems, the potential situations and tasks that arise during the life of a user interface become unpredictable. And if traditional, static design is the only available method for delivering these interfaces, responding to non-determinate situations becomes impossible. At the very least, adapting the interface in response to changes in task requirements, system configuration, user roles, or evolving technology or culture becomes difficult and costly.

For instance, consider the domain of building management. Even though the basics of day-to-day management are the same, individual installations are very different from each other, with different users, tasks, equipment, and requirements. Small buildings differ from large buildings, forced air systems differ from boiler systems, automated security installations differ from buildings staffed with human security guards. Large installations may employ large numbers of users, each assigned a well-specified role limited to one or two tasks required by the system; small installations may employ one user who is manager/operator/technician all in one.

This variability among systems in terms of configuration, criticality, tasks, roles, and security levels drives up the cost of traditional user interface methods considerably. In current systems, each visualization display that each user requires must be individually handcrafted, at a cost of up to 5% of the entire system.

We have been conducting large-scale research programs to address these issues, and believe that an appropriate and effective response is to develop systems that automatically design interfaces, responsive to the specific situation that holds at the particular time the interface is needed.

Dynamic Interaction Generation

Dynamic Interaction Generation (DIG) is a solution in which an automated reasoning system designs a user interface for a control system on demand and presents it on whatever display device is in use (Penner 1998). As a result, the interface is consistent, well-designed, and adapted to users' privileges, roles, and tasks; the objects of interest; the value and type of data displayed; and the hardware devices displaying the interface. As a user's interaction progresses, DIG continues to dynamically adapt the interface to these factors.

We have implemented a working DIG system in DIGBE (DIG for Building Environments), which converts a building management configuration database into a dynamic, adaptive interface for building managers. DIGBE currently uses the Honeywell Excel Building Supervisor (XBS) as the source of real-time operational data, as well as information about the domain objects present in the system. However, DIGBE's architecture (Figure 1) is modular, so that the system is neutral about its data sources. Efforts are currently underway to provide conduits to other real-time data sources and to automatically create configurations through discovery.

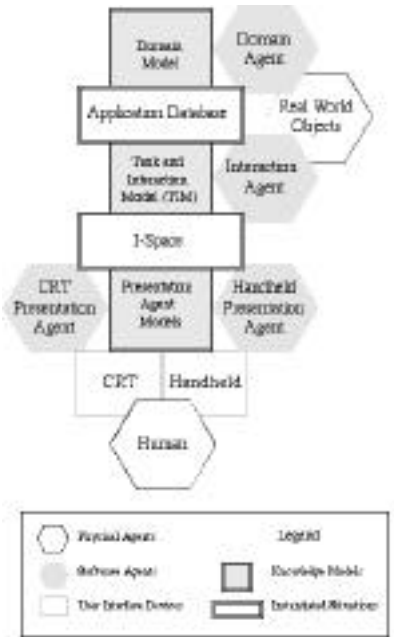

Figure 1: DIGBE Architecture

Architecture

The DIGBE architecture contains three basic knowledge structures, each manipulated by different types of software agents. A *domain agent* manages understanding of the relevant data in the real world. An *interaction design agent* performs automatic compositional interaction design. And a *presentation agent* determines how to present these interactions on the user's selected interaction device.

Each DIGBE agent maintains multiple models of the kinds of objects and actions it is interested in. DIGBE's knowledge models are:

- the *Domain Model*, used by the domain agent, which defines an ontology of the objects, relationships, actions, user roles, data roles, and data types in the domain
- the *Task and Interaction Model* (TIM), used by the interaction agent, which contains a multi-layer description of a compositional process for designing interactions to support applications and their tasks
- the *Presentation Agent Models*, used by presentation agents, which contain platform-specific widget libraries and platform-generic tables of heuristics and equivalencies, allowing conversions of the above interaction designs into working user interfaces.

DIGBE also contains two shared situation representations; the Application Database—which represents the current situation in the system—is shared by the domain agent and the interaction agent, and the I-Space—which represents the current interaction—is shared by the interaction and presentation agents.

Design Process

The basis of DIGBE's adaptive design capabilities is the TIM, which has five levels:

- *Applications*, like `BuildingManagement`, which are composed of tasks
- *Tasks*, like `Monitoring` or `Specifying`, which are composed of sub-tasks
- *Sub-tasks*, like `Plotting` or `Selection`, which are composed of elements
- *Element*s, like `Discrete Set` or `Value Over Time`, which are composed of primitives
- *Primitives*, like `Selector` and `Labeler`, which are the basic units of interaction

The TIM is a *self-composing productive system*. When an instance of a TIM object is created in the I-Space, that instance is responsible for adapting to the current situation and producing its own parts. Each object fine-tunes its sub-components for the specific context, based on appropriateness to the situation and the user. Using this process of self-composition, an entire interface—or only parts that have changed—can be created in real-time as needed. After an I-Space is fully composed, the interaction agent passes the interaction design to the appropriate presentation agent, which expresses the interaction using device-specific resources.

Benefits of Interaction Generation

When graphical user interfaces became practical in the 1980's, an emphasis on user interface management systems (UIMS) resulted in attempts to automate interface generation. Industry and university researchers generally concluded that user interface generation was too difficult (Szekely 1996), because it depends on human knowledge of task structures and domain requirements. In contrast, we hypothesize that the unpredictability of tasks and information available within emerging systems will only become more important, outweighing the difficulty and expense of defining and applying dynamic design knowledge. DIGBE demonstrates that dynamic interaction adaptation through separation of interface and application function is not only feasible, but possible as an automated real-time process, with minimal demand on domain semantics.

References

Penner, R. 1998. Automating User Interface Design. In Proceedings of Systems, Man, and Cybernetics 1998, San Diego.

Szekely, P. 1996. Retrospective and Challenges for Model-Based Interface Development. In Proceedings of CADUI '96, ed. J. Vanderdonckt. Namur, Belgium: Namur University Press.