DIGBE: Adaptive User Interface Automation

Robin R. Penner, Ph.D.
Erik S. Steinmetz

Honeywell Technology Center
3660 Technology Drive
Minneapolis, Minnesota 55418
penner@htc.honeywell.com

Abstract

We have developed an automated reasoning system, called Dynamic Interaction Generation (DIG), which automatically designs and dynamically presents adaptive user interfaces. We have demonstrated DIG in the domain of building management in the form of the DIGBE system. In this report, we focus on the mechanisms we employ to specialize dynamically generated interactions for the situational needs and constraints present in a building operations environment. These constraints are introduced through the control system and domain objects, the users and user roles, the tasks and subtasks that are required, and the need for both anchored and mobile interaction. To adapt the user interface that DIGBE designs for each user and situation, the system uses a combination of mechanisms including role based task composition and object specialization. It responds dynamically by constructing and maintaining real-time models of the system of interest and the user-system interaction, and provides device independence through the separation of interaction and presentation reasoning and the ability for multiple presentation agents to use a single interaction design.

Dynamic Interaction Generation

We have been investigating the types of knowledge, information processes, and models that are required to support adaptive dynamic design. In aid of this, we have been developing an assistant for dynamic interaction design, which we call DIG (Dynamic Interaction Generation; Penner, 1998), and applying it within the domain of collaborative systems management. Our basic requirements are to:

- minimize domain semantic requirements,
- maximize reuse of interaction design and presentation reasoning,
- minimize end user or design engineer configuration requirements,
- provide for complex constraints on individual functionality and data access, and
- provide device independence through the separation of design and presentation reasoning.

This research has resulted in the implementation of a set of modular automated reasoners that automatically and dynamically design and create user interfaces to large buildings and their associated control systems. The Dynamic Interaction Generator for Building Environments (DIGBE) is a fully functional system for managing the basic tasks of building management, including configuration, monitoring and control of security and environmental systems, management of users, and data analysis.

DIGBE

With no user interface code from a building management system (BMS) application, DIGBE generates the required navigation, monitoring, and data modification interactions required for several different classes of BMS users. Each user interface (UI) is presented as a coherent application, specialized when the user logs on to that user's particular role. A heating and ventilation (HVAC) technician, for example, would see only heating and ventilation information, and have access to a full range of data and data modification capabilities. A security guard, on the other hand, has access to very different tasks and limited control over a different set of information.

DIGBE *automatically* provides a well-designed, user specialized interface for each user. It also adapts *dynamically* as it supports the collaboration between the user and the system, by:

- automatically designing an appropriate application shell and required task interactions,
- dynamically specializing this interface based on the current user and domain situation, and
- interactively presenting the user interface for the interactions that are required on the user's selected device (CRT or handheld).

DIGBE Functional Components

The functional architecture of the DIGBE system is diagrammed in Figure 1. The components of DIGBE are three reasoning systems: Domain Reasoning, Interaction

Reasoning, and Presentation Reasoning. Each reasoning system is composed of a number of small executors (agents), each managing a knowledge structure. The agents initiate automated processes encapsulated in their knowledge structures to perform the functions in their area of expertise.

Agents send needs to other agents based on user inputs and changes in domain objects. The basic operation of a DIGBE agent is to react to a need from another agent by selecting an object from its model that is appropriate to the need, and telling this object to create or modify itself. Each object creates itself and its sub-components based on the current overall environment, and each sub-component, in turn, creates itself and its parts.

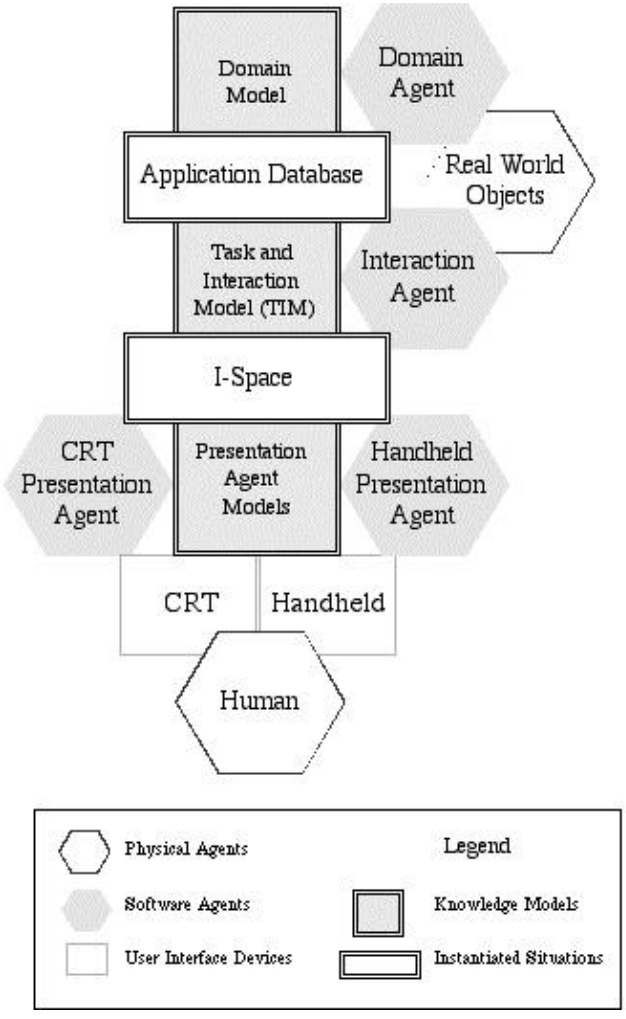

Figure 1. DIGBE Architecture

Each DIGBE agent has multiple models of the types of things and actions that it is interested in. It should be emphasized that the agents in DIGBE are actually little more than very simple model managers when separated from their knowledge structures, which define the semantics of the agent's area of expertise. The knowledge models in DIGBE are:

- the Domain Model, used by the Domain Agent, which defines an ontology of the objects, relationships, actions, user roles, data role, 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, and
- the Presentation Action Models, which contain platform-specific widget libraries and platform-generic tables of heuristics and equivalencies, allowing the Presentation Agents to convert the interactions (that the Interaction Agent designs) into user interfaces appropriate to the human and the interaction hardware.

Individual agents have different objects and processes in their models, as appropriate to the information they require and the operations they perform. They use various methods to determine which objects in their models to instantiate into their situation representations. DIGBE contains two shared situation representations; the Application Database (containing the representation of the current situation in the system) is shared by the Domain Agent and the Interaction Agent, and the I-Space (containing the representation of the current interaction) is shared by the Interaction Agent and the Presentation Agents.

Automatic Design

Each active DIGBE system automatically designs and presents itself, dynamically responds to changes in application objects, and tunes itself to the role and security properties of the user, the selected device, and the tasks that are active. The basis of these capabilities is the knowledge that DIGBE possesses about dynamic interaction generation.

DIGBE's Task and Interaction Model (TIM) represents the Interaction Agent's knowledge structures. Its purpose is to model the process of designing the interactions that are required to support user-system collaboration. TIM is hierarchical, describing applications, tasks, subtasks, elements, and interaction primitives as a constrained compositional system. Objects at each level are composed of objects in the next lower level, with the exact composition determined when the object is created or changed based on the constraints within and between each object.

The TIM model is a *self-composing productive system*. When an instance of a TIM object is created, that instance is responsible for adapting to the current situation and producing its own parts. Objects fine-tune their sub-components for the specific context, based on appropriateness to the situation and the user. Using this process of self-composition, and entire user interface (or only the parts that have changed) can be created in real time when needed. It is only necessary to tell a TIM Model

that a building environmental control operator user interface is required, and all the frames, navigation hierarchies, graphics and buttons are created automatically. The user interface that is presented is specialized dynamically to suit the user, the objects in the real world, the interaction devices, and the required application tasks.

Adaptation to User and Role

When a user logs in to DIGBE, the I-Space that is built to support their interactions with the system contains the tasks that that user requires for their defined role (Manager, Operator, Technician, etc.). The interaction situations for individual users are also constrained by the devices they are permitted to access (a user's *access level* in building management contexts) as well as by the systems they are interested in (security, safety, environmental, etc.). DIGBE considers all these required adaptations when it designs and presents the user interface.

An example of this is shown below. Figure 2 is a screen shot of a user interface presented to a user logged on as an operator of a large building, with interest in both security and environmental systems.

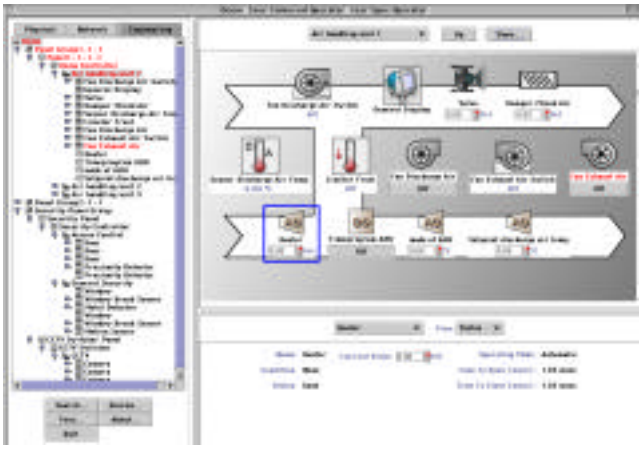

Figure 2. HVAC and Security Operator UI

Figure 3 presents the same situation, but with a user logged on as an HVAC technician. Note that the operator, above, has a much longer hierarchical list of accessible objects in the left panel, since that user is interested in both environmental and security system objects. Also, although each user receives the same type of screen layout, and their graphical monitoring and command capabilities are similar, they receive very different information when they interrogate objects.

In the situation shown in Figures 2 and 3, each user has selected the "Heater" (the object surrounded by the square in the lower left corner of the monitoring graphic). In both cases, the detailed information for the selected object is shown in the lower right frame of the display, but the contents and format of the information is different for the two different users.

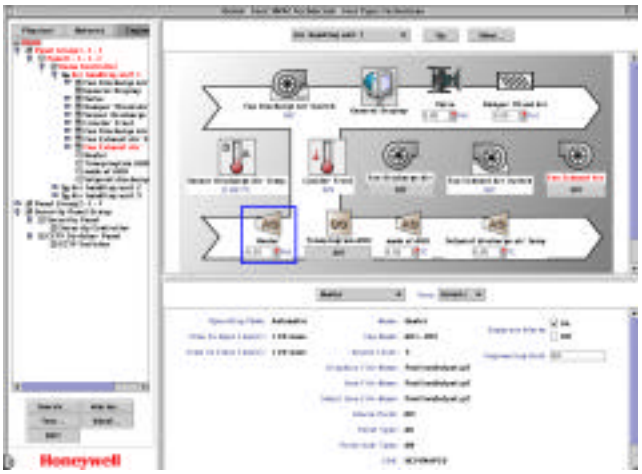

Figure 3. HVAC Technician UI

In the case of the operator, the information is appropriate to the tasks and role of this user, emphasizing simple controls and simple status information. The technician, on the other hand, receives diagnostic and performance information, and a much wider range of control and configuration options. In addition, the technician has a broader range of tasks, and each task has more capabilities than do those of an operator.

Adaptation to Information and Objects

When DIGBE creates or modifies a user interface, it not only adapts to the user and the task, but it adapts to the type and status of its application objects, as well. Each interaction is individually crafted to provide the best representation of the objects and information in the real world. This is accomplished in part through the representation of interaction objects as abstractly as possible in the TIM, using abstract classes that cannot be instantiated until they are specialized further.

An abstract element specializes itself by matching the interaction needs of the information it represents with the capabilities of its specializations to determine how to refine itself. This allows real time flexibility to define interactions, allowing the TIM to contain only one task definition that is flexible enough to allow different users to interact with many different types of objects, without advance knowledge of the objects themselves.

For example, the TIM defines each of the objects in the monitoring graphic as a composition with several standard parts. One of the parts is the value, which, for a commandable object, must be an active control allowing the user to change the value. There are many different types of objects, however, which need different types of user controls. The controllable fans are state switches, requiring detent pushbutton interfaces when displayed on a CRT, while Analog Output (AO) objects (like the selected Heater) are set by selecting a continuous variable (with its own individual limits and accuracy and units). AO objects

require “spinner” type interfaces, which increment or decrement in appropriate steps and contain the proper values and labels and limits.

Since it is impossible to tell in advance what type of objects will be present on any particular monitoring graphic, its components are defined abstractly. If an object on a monitoring graphic has data associated with it that fills a *value data role* for that object, it shows that data in its interaction by making a sub-component for the data. If the data also fills a *command role* for the object, this sub-component is defined as a generic type of interaction element called *Dynamic Information*.

When, for example, the heater object is building itself to provide elements that allow the user to command its value, it asks the abstract Dynamic Information object to create itself to represent the value of the heater in the application database. In the building management domain, a heater's current setting is represented as a continuous data object with particular limits and units. The Dynamic Information object selects the specialization that best represents continuous data objects, and sends the "create yourself" message on to that object. In the case of the heater, this object is a spinner interaction element, which creates itself appropriately with the high and low alarm and logical limits defined by the information it represents.

Conclusion

We have discussed two of the mechanisms that are used by the DIGBE system to create user interfaces that dynamically adapt to the user and the environment. DIGBE provides other adaptive mechanisms, such as utilizing multiple presentation agents to accommodate multiple hardware devices. Other features of DIGBE, including automated creation of domain roles and the use of domain interfaces to support domain independence, assure access to the semantic information required to support adaptation.

With DIGBE, we have demonstrated that dynamic creation of fully adaptive user interfaces is implementable within an object oriented component architecture. In the next phase of our work, we hope to demonstrate automatic discovery of domain information, extend the modeled domains, users, and applications to broader command and control situations, and add programmer's interfaces to allow user interface design specialists to refine the design knowledge possessed by the DIG agents.

References

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