

Automated Support for Human Decision and Control in Military Systems with Varying Levels of Autonomy

Robin R. Penner and Erik S. Steinmetz

Iterativity, Inc., 118 E. 26th St., Minneapolis, MN, 55404

robin@iterativity.com, erik@iterativity.com

Abstract From the perspective of the human commander of a complex automated process, an important factor that influences the quality of the command dialog is the ability of the user interface to provide information that allows the commander to perform situation assessment. Another important factor is the ability of the user interface to provide accessible and usable mechanisms to support communication between the human and the control system. We believe that much of the substrate for such human-machine dialog may be provided by a dynamic, model-based representation of the system components and processes, shared by all software and human agents participating in the control situation. In addition, we believe that the current method of providing user interactions to dynamic control systems, which involves static pre-design of all user interface components and interactions, is inadequate to meet the needs of multi-agent systems requiring varying levels of participant autonomy and varying levels of mixed-initiative control by human commanders, especially those in dynamically changing situations such as those present in a military battlefield. We have developed an approach that overcomes these limitations, using model-based productive reasoning about situations and interaction design, and are successfully applying this approach in the domain of military command and control of teams of semi-autonomous vehicles.

Key Words: User Interface Design, Model-Based Reasoning, Design Automation

INTRODUCTION

We have been investigating methods for supporting the human decision-maker who requires access to a full range of automation levels and a full spectrum of mixed-initiative control over complex automated systems. Our recent investigations center on the area of human command and control of groups of intelligent machines, with the aim of reducing the number of humans required. We are specifically targeting the domain of military command and control of squads of uninhabited aerial vehicles (UAVs). In a military situation, the commander provides the *command concept* (Builder, Banks, and Nordin, 1999) that drives the behavior of the hardware, software, and human systems involved in the situation; our interest is the facilitation of the command concept dialog between humans and machines.

This research addresses two important problems that stand in the way of producing good decision and control user interfaces. First, existing command and control systems do not represent the situation as a whole in ways that capture the semantic relationships between the situation components. This makes it difficult to present unified situational pictures to the

decision maker. Adequate situation awareness is a cornerstone of decision-making, and provides the basis for the formulation of the command concept by the human decision-maker.

Second, user interfaces to current control systems do not allow the user to fully interact with the system at varying levels of detail with the required flexibility, situational responsiveness, and adherence to good design principles that are required for an effective command dialog between human and automation. Any human command concept will be ineffective if it cannot be expressed to the hardware and software systems that will implement the concept. In addition, without access to the full range of system behaviors, and without adequate and responsive interaction design, the human will be unable to perform required activities at various levels of the control hierarchy. Not only will the automation be unable to understand the human, it will be impossible for the human to understand the actions of automated components or know how and when to intervene. Finally, any statically designed user interface will not be able to respond to unexpected situations or easily adapt to new functions or situation components.

We have found that the combination of a model-based approach to basic reasoning on situations, together with the ability to synthesize user interfaces based on the state of a situation, provides the ability to automatically design useful interaction, allowing the commander to successfully convey and oversee the implementation of a command concept that involves automated systems.

MODEL-BASED SITUATED REASONING

As part of the SHARED project, funded under the DARPA Mixed-Initiative Control of Automaton (MICA) program, we are developing a system to support commanders engaged in the command and control of fleets of UAVs. An example from a generated user interface is shown in Figure 1. Using a simulation as the source of situation information, the system builds a semantic model of the situation, designs and presents visualizations to support decision-making, calls planning reasoners as required, and provides interfaces to allow the commander to configure and control equipment, convey the command concept, and oversee mission execution.

Figure 2 illustrates the architecture of the SHARED system. On the left, a number of external planning agents are called as necessary by a central representation of the current situation. This situation representation is created and maintained under control of a **Situation Agent**. A set of agents (the **Interaction Agent** and the **Presentation Agent**) that design and present user interfaces are combined to form the Automated Interaction Designer (AID). AID dynamically produces user interfaces to the situation, as appropriate to the human user's current needs.

Object-oriented models provide the knowledge base for the situation agent and the AID agents. The **Domain Model**, under control of the situation agent, provides the semantic basis for interaction design, allowing control system independence and an object-oriented representation of control system components. The hierarchical, task-driven **Interaction Model**, driven by the interaction agent, provides the design knowledge required for automatic composition of appropriate tasks and abstract interactions, allowing dynamic application of the principles of usability engineering to support the design of the interactions between people and systems. The **Presentation Model**, driven by the presentation agent, possesses knowledge about how to select and format concrete user interfaces to express interaction designs, allowing hardware and operating system independence, and a clear separation between abstract interactions and concrete interfaces.

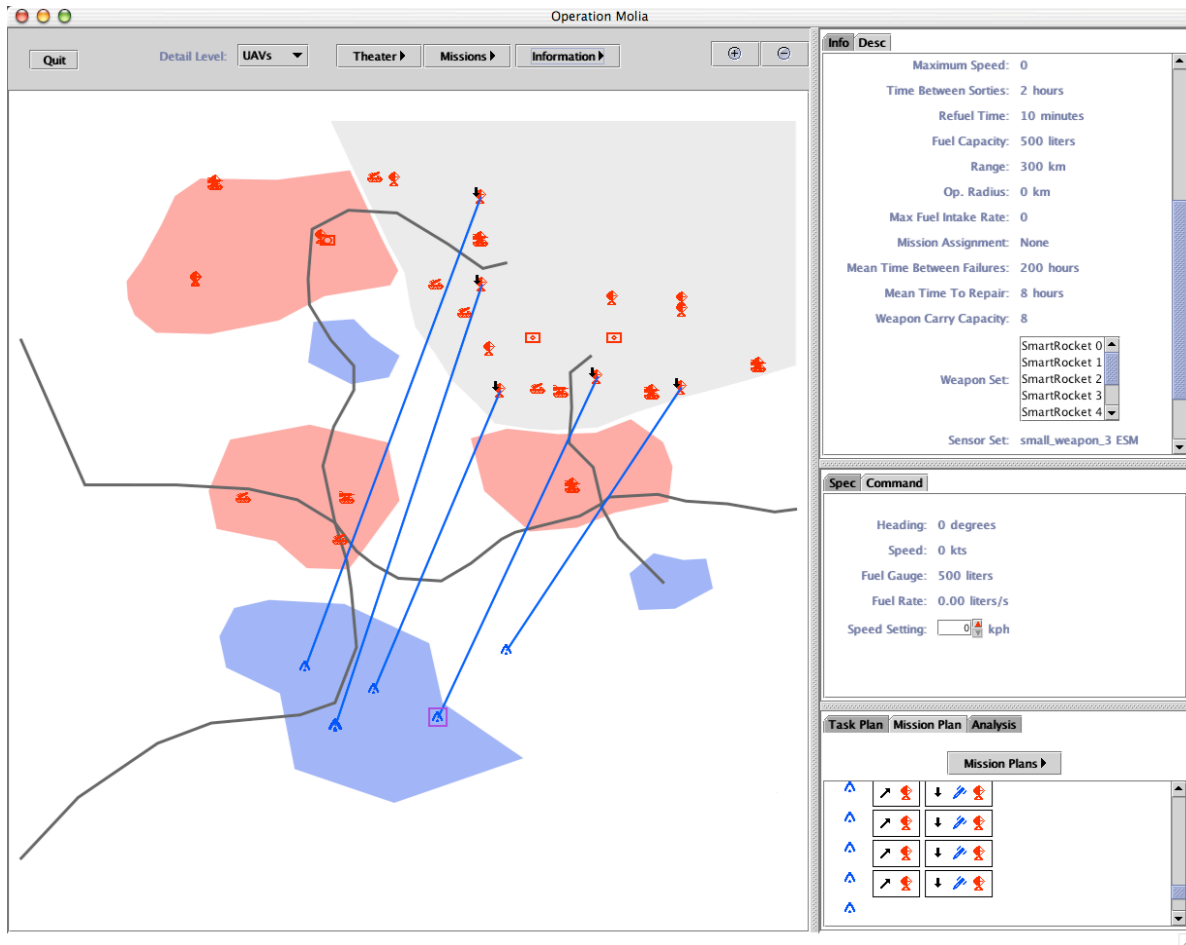


Figure 1. Example SHARED User Interface

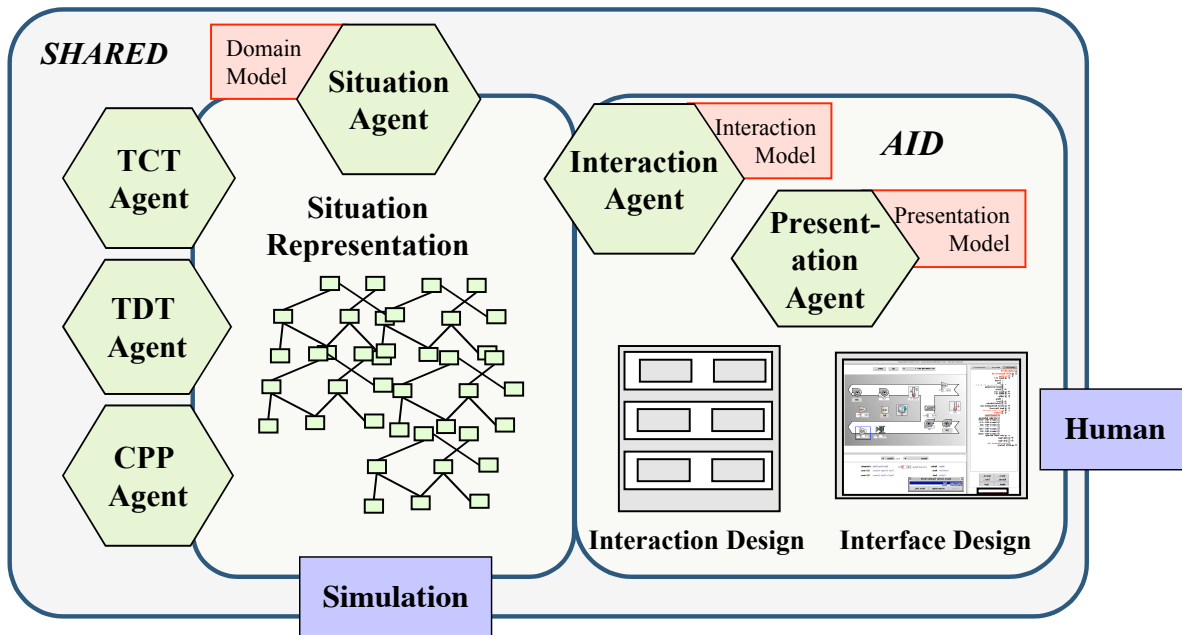


Figure 2. Model-Based Automated Interaction Design Architecture

Under the control of their respective software agents, each model is used as an exemplar database to produce dynamic models of the ongoing situation, the current interaction design, and the current presentation. An object-oriented **Situation Representation**, containing a representation of the system's situation in the real world, is created from the domain model. All interaction participants share the dynamic situation representation, ensuring shared knowledge for grounded collaboration. The interaction agent starts the design process for a particular user by creating an interaction object, which specializes itself using a compositional productive process to create an **Interaction Design**. The presentation agent (under the direction of the interaction agent) selects the appropriate presentation model for the currently accessed device (CRT/keyboard or handheld, in the current systems). It uses the objects in the presentation model as templates, selects and specializes them as necessary, and presents the Interaction Design as an **Interface Design**. As the user interacts with the situation through the generated user interface, or as the situation changes, the entire automated interaction design system continues to dynamically support the necessary user-system conversation.

Additional, non-model based reasoning modules are incorporated in the SHARED system. These include the **Team Composition and Tasking (TCT) Agent**, the **Team Dynamics and Tactics (TDT) Agent**, and the **Cooperative Path Planning (CPP) Agent**. These agents are capable of performing, respectively, task level planning involving creation, scheduling, and assignment of teams against missions, mission level planning involving assignments and scheduling of UAVs in teams against activities, and path planning for UAVs. The TCT, TDT, and CPP reasoning modules are invoked as necessary by situation representation objects (task processes, mission processes, and UAV controllers).

Situation Representation

The semantics of the situation that is the topic of the human-system dialog are contained in the situation representation. The situation representation is dynamically created from the objects defined in the domain model. The schema for the domain model is shown in Figure 3. As this diagram shows, the root object in a situation is the Operation Situation, which hierarchically contains the task situation that is being managed by the commander. In addition, the operation situation contains the places and groups that are participating in the operation. These types of objects are hierarchically composed of Systems, Equipment, and Actors. Situations (at all levels) also contain processes, which in turn are composed of activities. Activities may have associated plans, and may be assigned to specific Actors (the "performer" of the activity).

The domain model is a hierarchical organization of class types, with the classes organized into semantic networks and containment hierarchies when they are instantiated to match an object in the real world. The situation representation is a dynamic model of the entities currently participating in the situation, and includes software objects (of the types shown in the schema, as appropriate) that are semantically interconnected and which reflect the state of their referents in the real world. For example, an equipment object representing each UAV is created, with appropriate capable actions (Fly, Sense, Attack, etc.) and appropriate data (fuel consumption rate, current speed) and parts (path planning agent, communications system, throttle) for each UAV in the control system. In addition, the UAV might have a controller (an Actor, human or otherwise, that controls its behavior), and be part of a larger system or group, like a squad. That group is part of an operation, and the UAV's controller may be participating in any number of processes at any level, and may be planned to perform any number of activities in those processes.

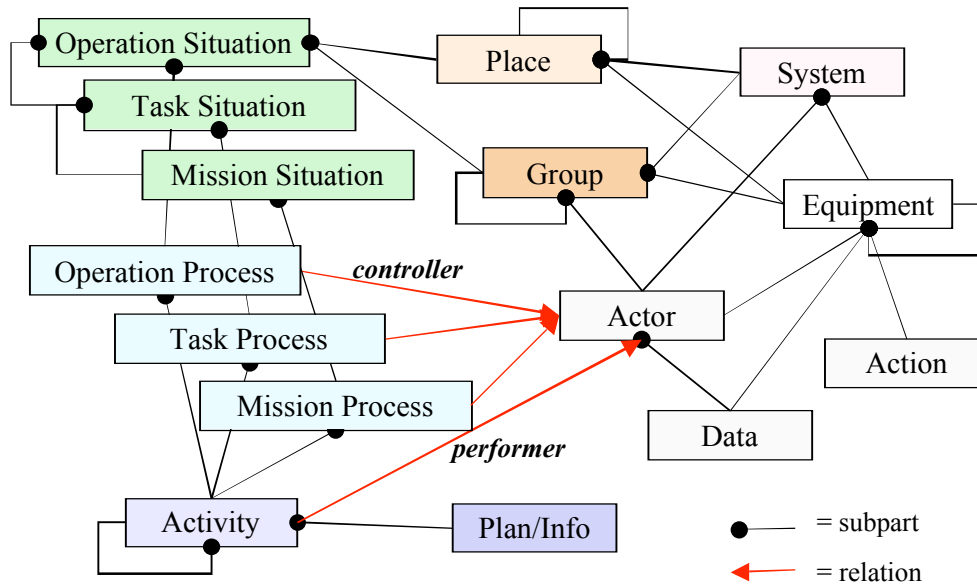


Figure 3. Schema for Domain Model

Figure 4 represents some of these relationships, and others available to a UAV through object-oriented inheritance. For example, each UAV has an endurance value, represented by a number class, a specialized type of data. Because UAV is a specialization of Aviation, it has another number representing its range, and also has associated sensors and weapons. Because Aviation is a specialization of Movable, all UAVs also have associated input and output classes (themselves made up of data) to represent such movable parameters as speed setting and heading.

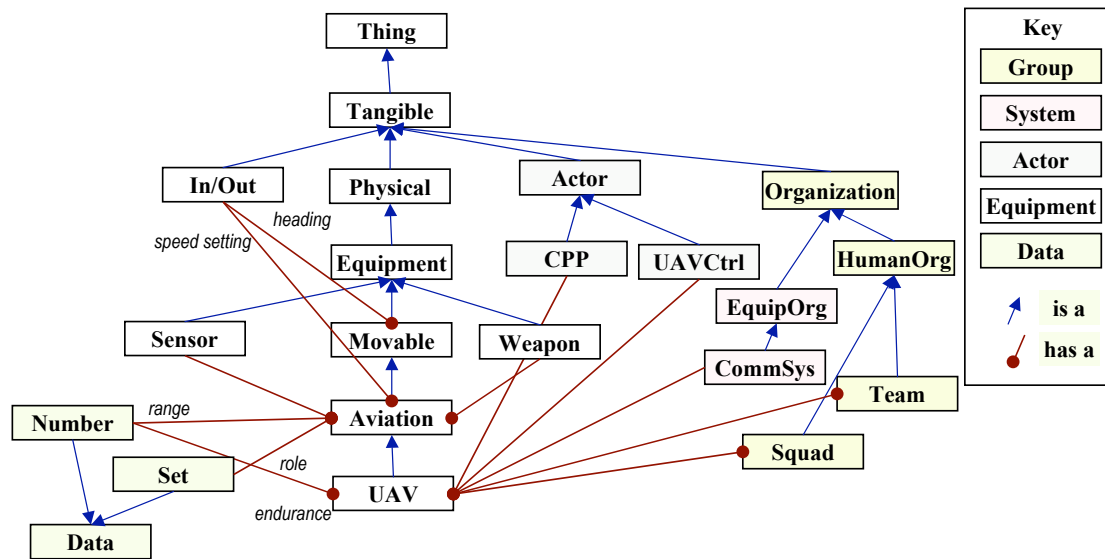


Figure 4. Example of Domain Model Inheritance

Activity Automation in Support of User Interface Creation

In its current implementation, SHARED obtains all information about the situation from an external simulator provided by Boeing. Initially, the system obtains the basic intelligence preparation of the battlefield (IPB) from the simulator, and uses the classes available in the domain model to create a unified picture of the situation according to the domain model schema.

An example of the contents of the situation representation is shown in Figure 5. This example shows the task situation that is created to represent the task that the commander is to perform with the allocated assets, and some detail of the basic task process that is created within the situation to enable the task to be completed. Also shown are the missions that are part of the task situation, with only the SEAD Mission Situation expanded. Because the IPB indicates that there are a number of Attack (with accompanying FlyTo activities), BDA, Find, and Identify targets, the appropriate activities for each target are created and added to the mission process.

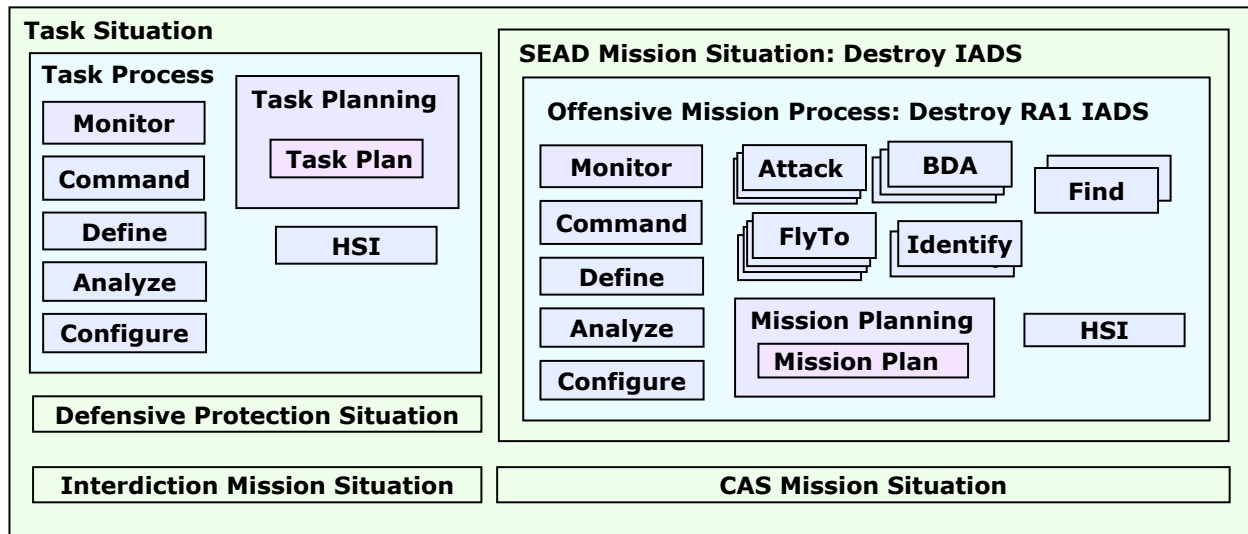


Figure 5. Example of Portions of a Situation Representation Created from IPB

Objects in the situation representation automatically react to the changes in the situation as indicated by simulator values, and objects and context are created and destroyed as necessary to provide a semantically rich picture of the actual situation. The situation model actively and dynamically performs target evaluation, creating missions and mission targets as necessary, for assignment by external agents like the TCT and TDT. Each situation type knows what sort of targets needs to be included in a process, subject to commander's guidance, objectives, and ROE.

The heuristics for target identification and classification are defined in the various mission situations, which consider each of the possible targets present in the area of interest to that mission (e.g., red area 1). An **Attack Activity** is created in an area's mission process for a given Mission Situation for each of the enemy objects that are considered "attack targets". Objects are considered attack targets if they are:

- 1) identified at the identification certainty level defined for the task, normally 90% certainty,
- 2) of the specified type according to default or commander's guidance, and
- 3) located in the area according to the ROE for that system

For example, no object can be attacked in a restricted area, any system designated as TCT ROE can be attacked wherever it is encountered, and a system with ROE set to KZ can only be a target if located within a kill zone.

A **DamageAssessment Activity** is created in each mission process for each Attack Activity, targeted at the attack target. An **Identify Activity** is created for each of those red objects in the area that do not meet the identification certainty rule, and is targeted at a red unit that has less

than 90% certainty (or the target of a “find” activity that has been “found” and now needs identification). A **Find Activity** is created inside each offensive mission process for each target that doesn’t meet the identity certainty criteria. In defensive mission processes, a **Protect Activity** will be created for each of the approaches specified for the defended blue area. **FlyTo Activities**, with the goal of the attack, search, or identify activities, are created for each Attack, Search, and Identify Activity.

Thus, the situation model productively performs the initial target planning for the situation. During planning, a mission-planning reasoner such as the TDT could assign multiple assets to a specific target through the creation and assignment of additional activities. The **ReFuel Activity** is currently not scheduled by default, but is added by the TDT agent along with the appropriate FlyTo activity as determined by that reasoner. The **Jam Activity** is also currently not scheduled by default, and is also added by the TDT agent as necessary.

The situation also automatically assigns actors to perform each activity necessary in the situation. Each actor in the situation has a list of capabilities, which are the type of activities that they can perform. The types of Activity Objects currently defined in SHARED are shown in Figure 6. A UAV Controller, for example, is capable of Refueling, and the preferred Actor for a Refueling activity is a UAV Controller. Other actors may also be capable of activities that are nominally performed by others. For example, although a Team Composition and Tasking Agent is the preferred actor to perform the Task Planning Activity, human actors are also capable of performing that and nearly all other available activities. Therefore, when an automated actor is unable or unavailable to perform assigned activities, these activities are automatically assigned to the human.

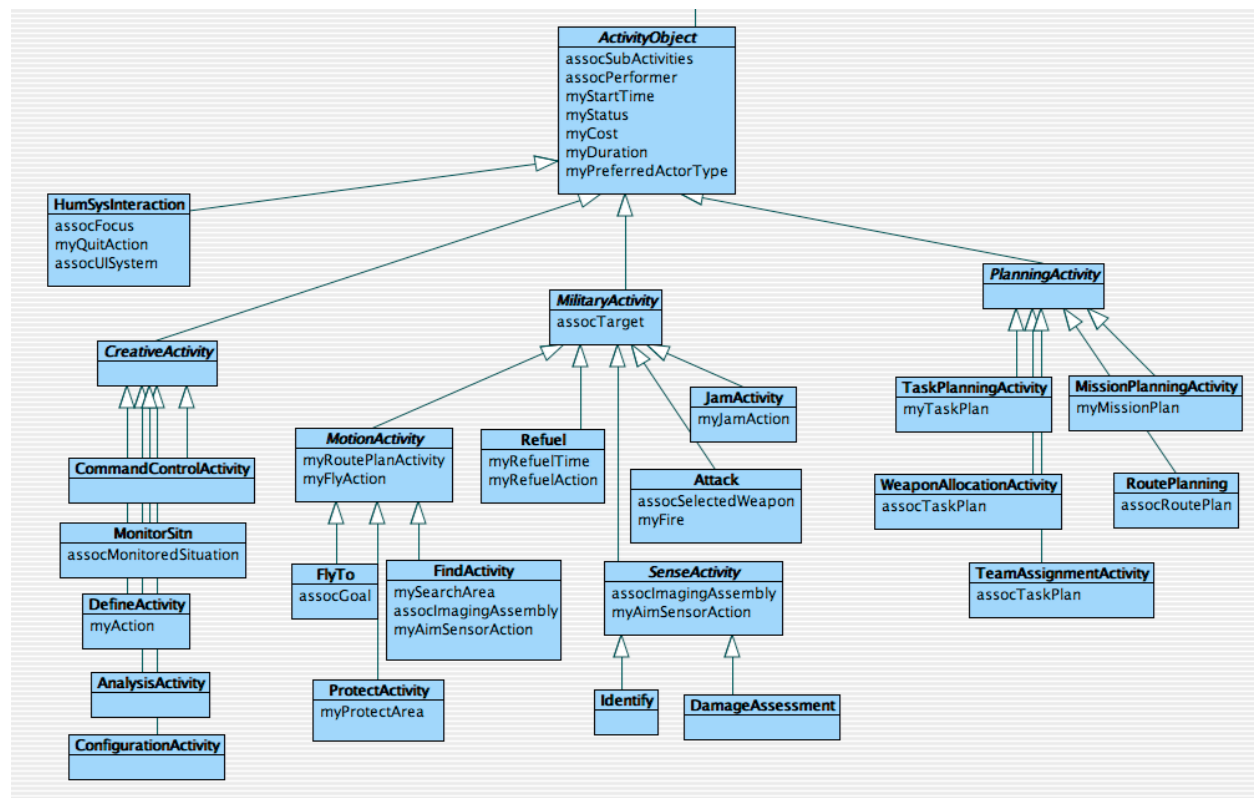


Figure 6. Activities Defined in SHARED

Figure 7 shows the actors defined in the SHARED system. Arrows with open heads (pointing upwards) represent inheritance; automation is a type of actor, and CPP is a type of automation. Each actor has an attribute that defines its capabilities in terms of the activities it is capable of performing, and each activity has an attribute that defines its preferred type of actor. For example, the preferred actor for military activities is a UAV Controller. Humans are the only actors in the system to be assigned the set of Creative Activities: Command and Control, Monitoring, Defining, Analysis, and Configuration. Military Activities are normally assigned to UAVs, Planning Activities to external planners, and Human-System Interaction generation activities to AID.

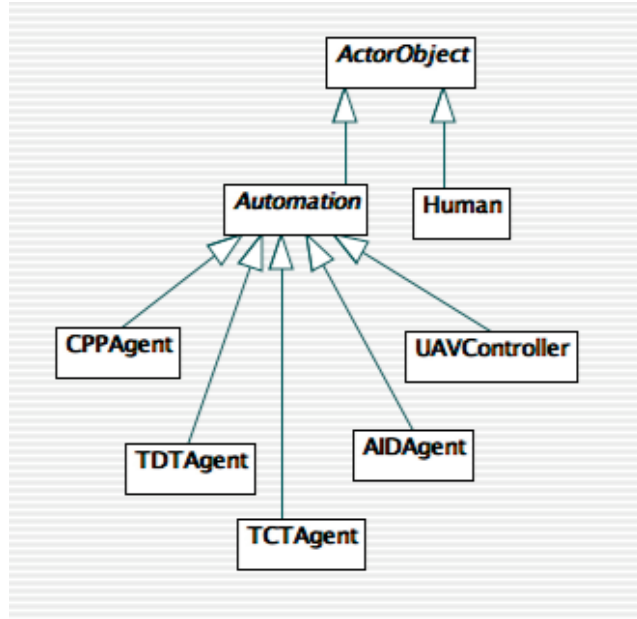


Figure 7. Actors Available in the SHARED System

Because the AID system is responsible for providing interaction support for all of the activities assigned to the human actor in the situation, the Interaction Model automatically creates views for each of the creative activities, and each interaction level is filled with the objects that appear in the situation. Each view in AID is defined in terms of the activities it facilitates; the views are automatically selected to match the activity needs, and each view automatically fills itself based on the status of the situation. Because the human commander is defined as being capable of any activity, all the activities present in the situation are automatically available to him through the user interface, although these “backup” activities are not given the prominence that is given to the activities assigned to the commander. An automation level is available to the commander at any time if he chooses to intervene, or if an external reasoner becomes disabled or is unavailable. In this way, the full spectrum of autonomy is supported, and the commander has full control of any component at any time.

AUTOMATED INTERFACE DESIGN SYNTHESIS

Research in the last several decades has contributed to the development of a solid practice of usability engineering, and has clearly demonstrated that the design of the user interface to an automated system determines the ability of humans to use it (see, for example, Landauer, 1995 and Nielsen, 1993). Processes and artifacts for performing user interface design have been

successfully developed and are in widespread use (e.g., Galitz, 1999; Mandel, 1997; Mayhew, 1999; Rosson & Carroll, 2002), and integrated user interface design and testing software is readily available.

In practice, however, fully usable systems often remain both undefined and unrealized, especially in the area of user interfaces to digital control systems. Uniformly good command and control interfaces have been difficult to achieve for many reasons, including the need for specialists to perform the design work, rapidly changing technology, increasing interaction options, difficulty quantifying procedures and results, and the difficulty inherent in producing and adopting useful specifications, standards, or guidelines. In addition, when standards and processes do exist, there is often little management support for their application by software engineers involved in real-world development efforts. In most cases, user interface design is not considered an integral part of the development process until late in the cycle.

These general problems with the delivery of usable systems are exacerbated in the various domains based on digital process control. In applications like factory control, building systems management, flight systems control, and operations management, user interfaces are usually individually developed for each application, installation, process, and, often, each user. This is an expensive approach, requiring skilled human labor. In addition, such systems pose a number of problems that may result in less than optimal user interface designs.

First, each digital control system is unique. Each system, even those with the same set of functions, has a different set of components, organizations, tasks, users, computing systems, control programs, and interaction devices. Second, these individual features change over time, sometimes very quickly, and all evolve with the advances in digital control technology. This frequently necessitates changes in at least the user interface portions of the control software, and often requires completely new user interfaces to be developed, installed, and mastered. Third, the users and tasks in this environment are diverse, and multiple separate software applications are required in order to support them. Even with proper usability engineering applied to each application, user interfaces to various systems and functions will differ, and each site that needs a particular application will require manual user interface specialization.

In response to these problems, we have been involved in a long-term research and development program to reduce the cost and improve the usability of digital control interfaces (Penner, 1992, 1993, 1994, 1996, 1999; Penner & Soken, 1993, Penner & Steinmetz, 2000, 2002). Initially, we limited our research domain to the relatively simple one of building management. We have continued the program in the more complicated domain of military operations planning and control, and have combined it in the SHARED program with the compositional model-based situation representation to provide fully situated dynamic user interfaces. The result of this research has been the design and implementation of several online model-based systems that create, display, and modify all user interfaces that are required by each user of a particular type of digital control system. All aspects of the user interface are dynamically created on demand, specialized for the situation, the specific users, and the interface devices they choose to use.

The Automated Interaction Design (AID) approach has allowed us to explore the allocation of the various functions that are required to drive automated interface generation, as well as to experiment with mechanisms for arranging each model and accomplishing each required function. Systems built using the AID approach allocate knowledge about domain semantics, interaction design, and interface presentation into separate, collaborating models. Each model

contains knowledge about a particular aspect of interface design, with the details distributed among hierarchically organized objects. The system uses this knowledge to dynamically create and manage all the user interfaces that are needed by any user.

The AID system is able to provide specialized user interfaces for each situation and user because it uses internal models of good design (as we currently understand them) to create and tune each interface. Responsibility for the various aspects of an interface design is distributed into different models that collaborate to produce the user interfaces. As Figure 2, above, illustrates, using the AID approach, the functions of situation representation, interaction design, and interface design are partitioned into three different object-oriented components, each with particular responsibilities in the collaborative design effort.

Interaction Modeling

The automated design capabilities of AID stem from the knowledge that the interaction model possesses about interaction design, and the ability to use the semantic information in the situation representation to drive the compositional design process. The interaction model is organized as a (hierarchical) cellular automaton driven by local behavior rules and affordance constraints (Penner & Steinmetz, 2002). Its purpose is to model the process of designing the interactions between the user and the information contained in the current situation.

The interaction model in AID is made up of three hierarchical levels, each composed of objects from the next lower level:

- *Views* (e.g., Monitoring), composed of elements;
- *Elements* (e.g., State), composed of primitives; and
- *Primitives* (e.g., Value), the basic units of an interaction.

Figure 8 illustrates the inter-relationship of the classes in the AID interaction model. The interaction model in AID is a *self-composing productive system*. When an instance of any interaction object is created for a particular user and a particular domain object (the *referent* of the interaction object), that instance is responsible for adapting to the communication needs of the referent and the user. The interaction model uses internal rules and constraints based on current practice in task analysis, requirements management, and interaction design, as appropriate to the responsibilities of each object. Interaction objects also select their sub-components for the specific context, based on appropriateness to the situation and the user. Each subcomponent then tunes itself by selecting and tuning its own subcomponents. Using this process of self-composition, an entire interaction (or only the parts that need to be changed) is created or modified in real time when needed. Each interaction design is specialized dynamically to suit the user, the objects in the real world, the interaction devices, and the required tasks. In use, each dynamically responds to user input (or changes in the system) by redesigning specific parts of itself, as needed.

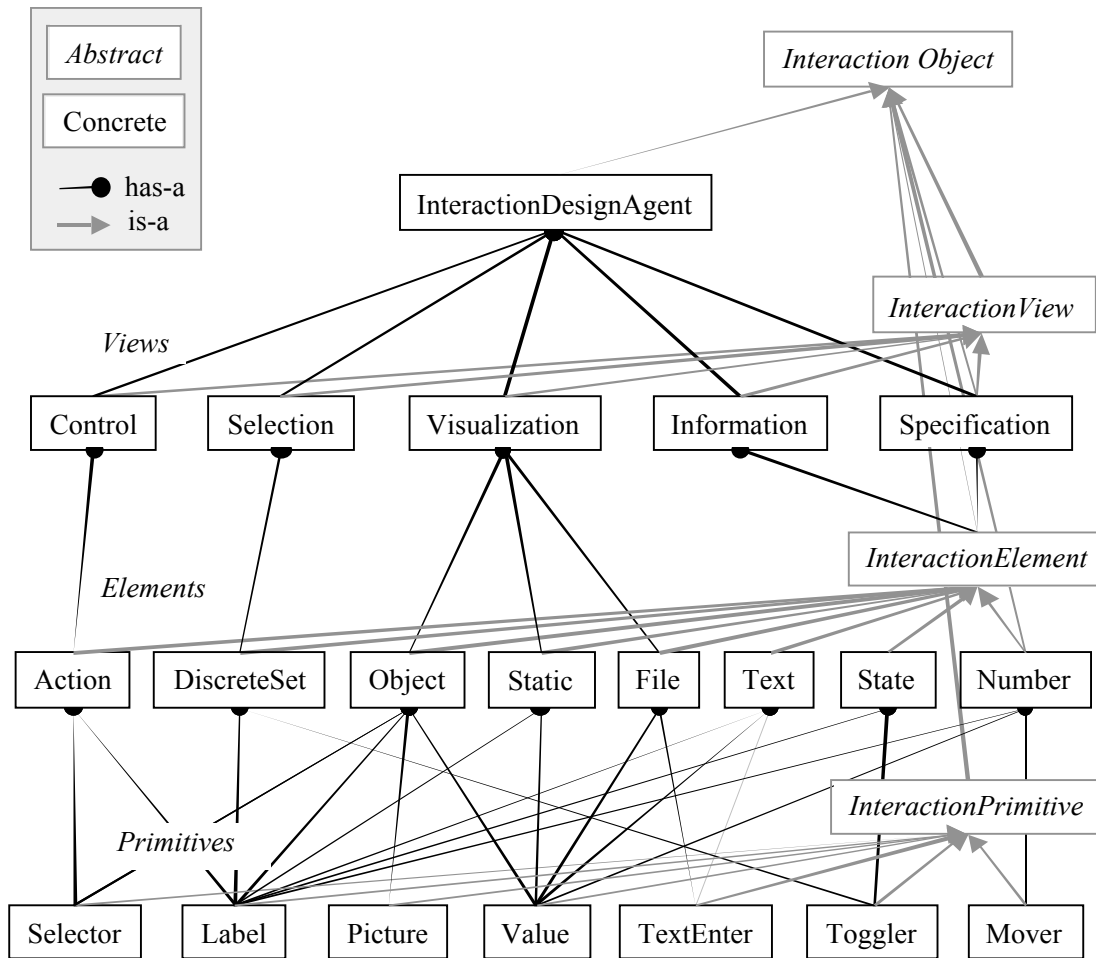


Figure 8. AID Interaction Model Overview

Figure 9 shows an example of the design of a Command View for a UAV, along with its component elements and the primitives for two representative elements. Each data item associated with the UAV that has a usage role of value or setter is created as a particular type of element, chosen to match the representation role of the data it represents. The situation object “Speed Setting”, for example, is a continuous data object, assigned the setter usage role when it was created as part of the UAV (since it provides an interaction that is used to set a speed). Continuous data objects have the “number” representation role. Because the interaction element that also has a number representation role is the Number Element, the Information View creates an instantiation of a number element into the interaction design, to represent the interaction with the speed setting of the UAV.

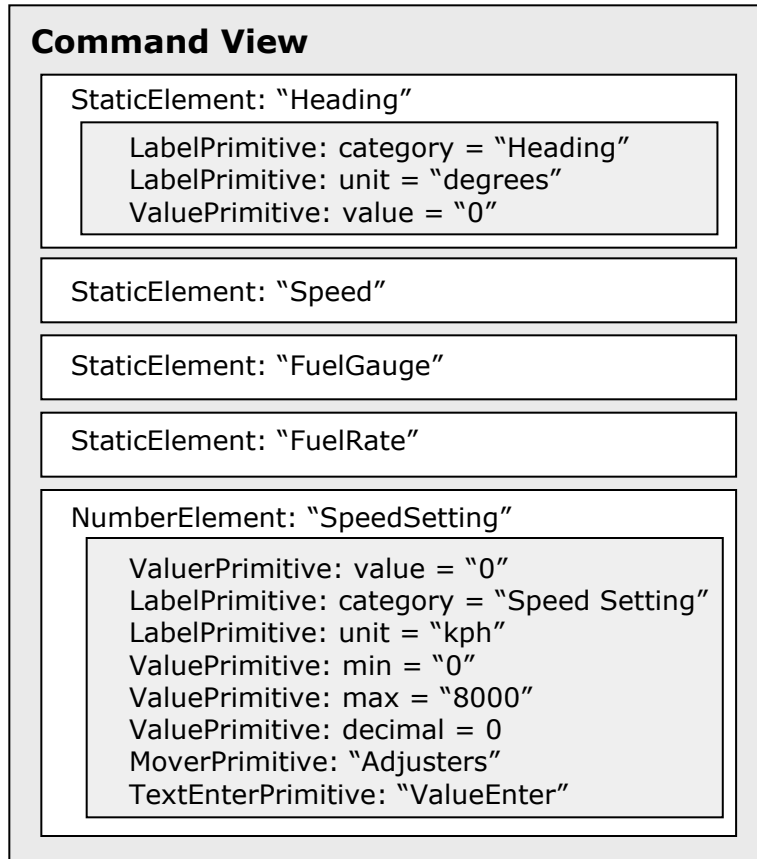


Figure 9. Partial Interaction Design for an Information View Referencing a UAV

Interaction elements have a pre-defined set of primitives, which are selected if the parts of an element they represent are present in the situation. The interaction element selects the appropriate primitives for the data object that is its referent, and fills the properties of the primitives with the appropriate values from the domain object. In Figure 9, there are two elements shown expanded into primitives; the first (a static element) and the last (the number element representing the speed setting). For a speed setting, the number element contains a number of primitives, including two label primitives providing the name of the category (in this case, "Speed Setting") and the units ("mph"). Four value primitives represent the current value (200), the decimal places (0), the maximum speed value (the maximum speed for this UAV), and the minimum speed value. A text enter primitive provides the ability to enter a new value, and a set of mover primitives allows the value to be scrolled. Each of these primitives has an appropriate referent in the domain; for example, the referent of the category label primitive is the object name data object for the referent of the number element.

Presentation Model

The presentation model is separate from the interaction model and the interaction design, in order to provide adaptation to multiple types of user interface devices, and to separate user requirements from the physical satisfaction of these requirements. Each type of device has a defined Presentation Model; the appropriate model is selected based on the device currently in use. The Presentation Model converts the interaction design into a presentation by selecting user interface components like windows (to enclose the interaction), frames (for views), and widgets

(for elements), and specializing them according to the parameters of the objects in the Interaction Design.

In addition to the device-specific knowledge about translation from interaction objects to presentation objects, each presentation model also contains a number of heuristics that specify how to select and specialize presentation objects, and how to code and present the interaction object information (for example, how to line up widgets, add appropriate status coding, and represent objects graphically).

Figure 10 presents a portion of an AID user interface design for the interaction design shown in Figure 9.

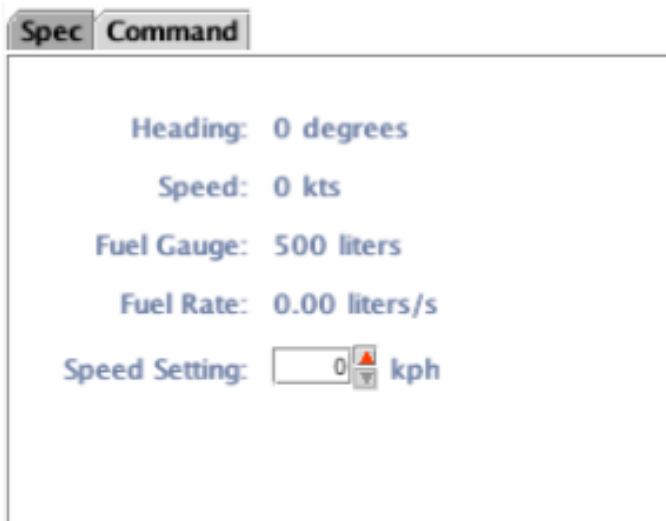


Figure 10. Example User Interface Design for Command Display with UAV Referent

In this example, the number element in Figure 9 (the speed setting for the UAV) is represented as a number spinner, which is the corresponding presentation component for a number element in the Presentation Model for a standard computer console. The category label primitive “Speed Setting” is used to label the number spinner, a colon is added to the label, and the entire label is aligned with other interactors in the presentation by the heuristics included in the number spinner presentation object. A text box represents the text enter primitive, and up/down arrows the movers. The high and low limits for the speed setting are provided by the maximum and minimum value primitives, and the text box and the movers are set to accept only values within those bounds. Finally, the actual value, contained in a value primitive, is placed in the text box.

Another example is shown in Figure 11. In this case, the situation model has assigned the activity of analysis to the human commander. Because the commander has selected to see the progress of all missions, the information about each mission that conveys evaluation information is selected. Because a Kiviati graph (the type of graph shown in Figure 11) is most appropriate to show comparison between a desired state and an actual state, that type of presentation is selected, created, and filled with the appropriate values from each mission.

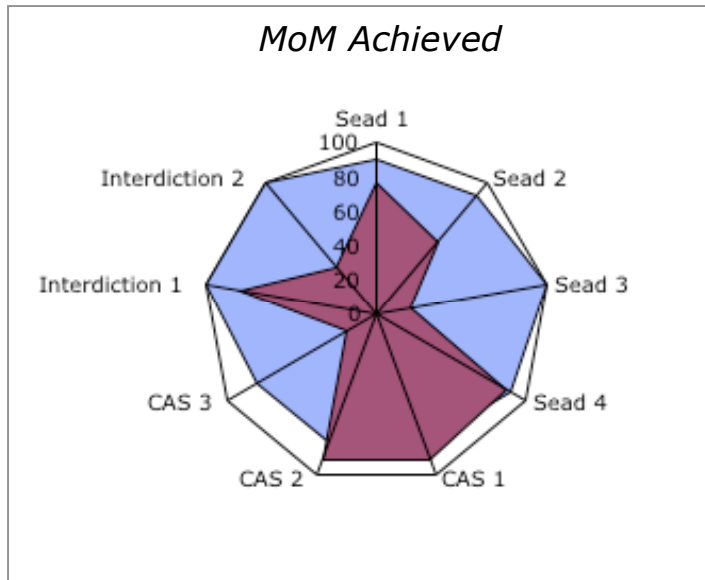


Figure 11. Analysis Display Example

BENEFITS OF MODEL-BASED AUTOMATION

“The design decision that has the most profound effect on the human-machine sciences is degree of autonomy” (NRC, 2000, p. 31). In C2 systems like those involved in controlling multiple teams of automata, user interface demands arise from the fact that the system to be controlled has a number of automated components, and those automated components have varying degrees of autonomy (both statically and dynamically). A model-based reasoning approach provides an ideal mechanism for supporting the human commander in the specification, monitoring, and modification of that commander’s command concept in such a system, because the interaction is generated based on the model-based automation reflecting the changing situation.

The use of productive model-based reasoning for situation representation and user interface design automation combine into an effective way to provide both situationally relevant interactions as well as well-designed and usable interactions. Because the situation representation is dynamic, compositional, and hierarchical, a fully specified semantic structure is available at any level. The situation as available to any agent in the system is guaranteed to be the same as that available to any other agent; in SHARED, the situation representation is the basis of the actions of such diverse agents as an automated user interface design reasoner as well as game theoretic planning reasoners. Because the interaction model is productive and compositional, and based on appropriately allocated capabilities and heuristics, the commander is provided with a user interface that allows situation assessment at any level. AID provides flexible and timely command concept dialogs with subsystems with variable levels of automation, providing well-designed and accessible interactions that accurately reflect the status of the objects and situations in the real world.

ACKNOWLEDGEMENTS

The work on AID is supported by DARPA and AFRL under contract F33615-01-C-3151.

REFERENCES

Builder, Carl H., Bankes, Steven C., and Nordin, Richard (1999) Command Concepts: A Theory Derived from the Practice of Command and Control. Washington, D.C.: National Defense Research Institute/RAND.

- Galitz, W. (1997) *The Essential Guide to User Interface Design*. New York: Wiley.
- Landauer, T. (1995) *The Trouble with Computers*. Cambridge, Mass.: MIT Press.
- Mandel, T. (1997). *The Elements of User Interface Design*. New York: Wiley.
- Mayhew, D. (1999) *The Usability Engineering Lifecycle*. San Francisco: Morgan Kaufman Publishers.
- National Research Council (2000) *Uninhabited Air Vehicles: Enabling Science for Military Systems*. Washington, D.C.: National Academy Press.
- Nielsen, J. (1993) *Usability Engineering*. Boston: Academic Press.
- Penner, R. (1992) *Consistent Honeywell Interface Design Concept for Building Management*. Minneapolis, MN: Honeywell Inc., Sensor and System Development Center, 1992.
- Penner, R. (1993) Developing the process control interface. *Engineering for Human Computer Interaction*, Elsevier Scientific Press, 317-334.
- Penner, R. (1994) Multimedia interfaces for process control. *Proceedings of the Energy-Sources Technology Conference, ASME petroleum Division*, 375-383.
- Penner, R. (1996). Multi-agent societies for collaborative interaction. *Proceedings of the 1996 40th Annual Meeting of the Human Factors and Ergonomics Society, Part 2 of 2*. Philadelphia, Pennsylvania, Sept. 2, 762-768.
- Penner, R. (1999) DIGBE: Adaptive user interface automation. *AAAI Spring Symposium*, 98-101.
- Penner, R. and Soken, N. (1993) Consistent Honeywell Interface: Tools for developers. *Scientific Honeyweller*, 106-109.
- Penner, R. and Steinmetz, E. (2000) DIGBE: Adaptive user interface automation. *AAAI Spring Symposium*, Stanford, CA, 98-101.
- Penner, R. and Steinmetz, E. (2002) Model-based automation of the design of user interfaces to digital control systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*. Vol 32, No. 1, January, 41-49.
- Rosson, M and Carroll, J. (2002) *Usability Engineering*. San Francisco, Morgan Kaufmann Publishers.