

Automated Interaction Design for Command and Control of Military Situations

Robin R. Penner

Iterativity, Inc.
118 E. 26th St., #201
Minneapolis, MN 55404 USA
+1 612 871 4514
robin@iterativity.com

Erik S. Steinmetz

Iterativity, Inc.
118 E. 26th St., #201
Minneapolis, MN 55404 USA
+1 612 871 4514
erik@iterativity.com

ABSTRACT

We will demonstrate the SHARED software, which contains an implementation of the Automated Interaction Design (AID) approach to dynamic creation of user interfaces. AID uses multiple agents, multiple models, and productive compositional processes to generate need-based user interfaces within a complex control domain. In addition to demonstrating operational software that responds to military interaction needs, we will present details of the underlying models and operations that support user interface generation in this domain.

Keywords

Automated design, model-based reasoning, dynamic user interface, domain semantics

INTRODUCTION

As part of the SHARED project, funded under the DARPA Mixed-Initiative Control of Automats (MICA) program, we developed software to support commanders who are engaged in the command and control of fleets of unmanned aerial vehicles (UAVs). MICA research emphasized mixed-initiative control of teams of automated systems, with the integration of numerous collaborative planning agents who operate with variable autonomy. Details of the approach and the architecture are given in [1], [2], [3] and [4].

Using a simulation as the source of situation information, SHARED 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 evolving command concept, and oversee mission execution.

COPYRIGHT IS HELD BY THE AUTHOR/OWNER(S).

IUI'04, JAN. 13-16, 2004, MADEIRA, FUNCHAL, PORTUGAL.

ACM 1-58113-815-6/04/0001.

SHARED PERFORMANCE

The architecture of the SHARED system is shown in Figure 1.

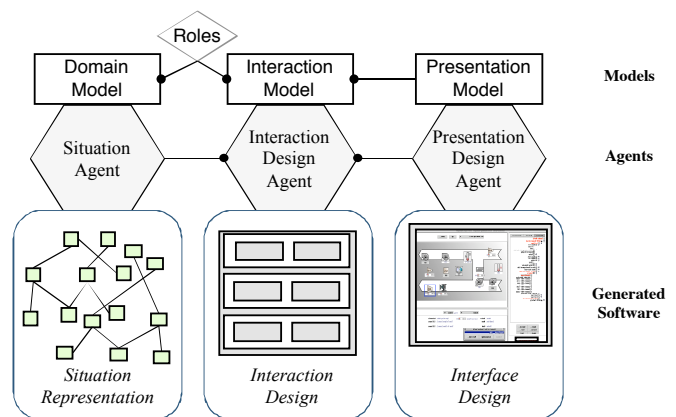


Figure 1. Architecture of SHARED

The steps in running the software begin with the commander starting the SHARED application. This causes the situation agent to connect to a simulation (or other data source) and collect information about the entities in the current battlespace. Based on this information, it uses exemplars from a domain model to create and maintain a semantically rich representation of the situation throughout the life of the active software.

After a complete situation representation has been formed, objects in the situation that need planning assistance call on any available external planning agents to produce the plans they require. Finally, because there is a human user in the situation representation who has responsibility for the command and control of the battlespace situation, the Automated Interaction Design (AID) module is called to build and maintain a user interface to the system.

The sequence diagram for the AID portion of the SHARED software is shown in Figure 2.

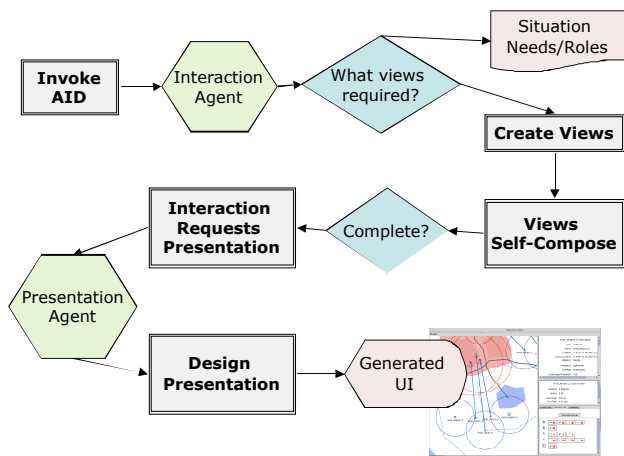


Figure 2. AID Sequence of Operations

The interaction agent creates a high level object to represent the interaction with the user. This object then composes subparts of itself (views) based on the activities, responsibilities, and capabilities of the human user. The “needs” are embodied in the objects, relationships, and roles present in the situation. To do this without dependence on prior knowledge of the contents of the situation, the domain model and the interaction model share a set of static roles available in a separate role model. Once the high level interaction has determined which views are necessary to support the user’s activities, it creates one or more views of each required type. These views then *self-compose* themselves, selecting and representing their required contents.

All objects in the interaction model (and many in the domain model) are productively self-composing. When created, they create their own subcomponents based on factors such as the context or the roles of the things in the domain that need to be represented in the interaction. Each view selects the required information about the situation by selecting the appropriate data associated with the object being represented. For example, specification views select the data associated with an object that expresses parameters, like heading or speed, while information views select information that expresses identity and description, like name and color.

Once the data that need to be represented have been selected, interaction elements to represent each piece of information are created, based on the format of the data. For example, the number element is chosen to represent the speed setting because it is both changeable under the context of a specification interaction and its data format is a continuous number. Each applicable data property of this speed setting is represented as part of the number element, using the appropriate primitives from the interaction model; for example, value primitives are used to represent current value, minimum value, maximum value, and the decimal places, while a text enter primitive is used to represent the editable setting value.

When the interaction design is complete, an agent specializing in presenting interactions to humans as concrete presentations is called. The presentation agent uses heuristics and best practices rules to group data, line up widgets as appropriate, and color or otherwise code the information for human consumption, and it dynamically manages translating user inputs back to the interaction design. As the user interacts with the system, the situation representation, the interaction design, and the presentation design all remain dynamically connected. This allows AID to automatically continue to meet the changing needs of the user. A screenshot from SHARED is shown in Figure 3.

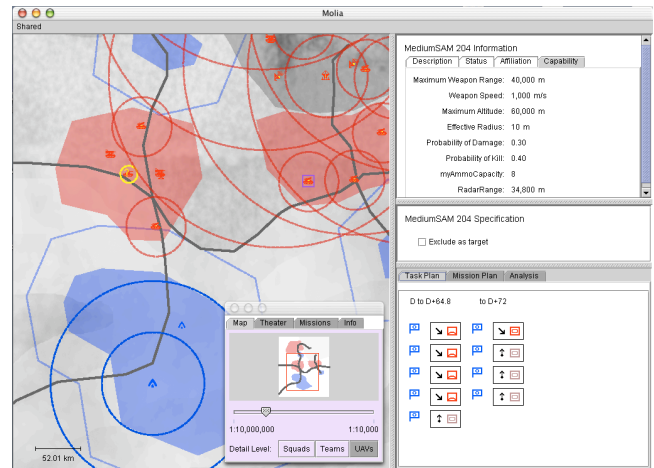


Figure 3. SHARED Example Display

AID automates, using a model-based productive process, the composition of user interfaces to meet user needs. The use of AID in SHARED, a fully operational software application for command and control of a complex military application, demonstrates the applicability of this approach.

ACKNOWLEDGEMENTS

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

REFERENCES

1. Penner, R. and Steinmetz, E. (2002a) 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.
2. Penner, R. and Steinmetz, E. (2002b) DIGBE: Online model-based design automation. Kolski and Vanderdonckt, eds., *Computer-Aided Design of User Interfaces III*, Kluwer, pp 179-192.
3. Penner, R. and Steinmetz, E. (2003) Implementation of automated interaction design with collaborative models. *Interacting with Computers*, Vol. 15, 367-385.
4. Penner, R. and Steinmetz, E. (2003) Automated Support for Human Mixed-Initiative Decision and Control. *Proceedings of the Conference on Decision and Control*, December.