AIAA-94-1249-CP

THE GROUND VEHICLE MANAGER'S ASSOCIATE

Gary R. Edwards, Robert H. Burnard, William L. Bewley, and Bruce L. Bullock
ISX Corporation
4353 Park Terrace Drive, Westlake Village, California 91361
Phone: 818-706-2020 Email: gedwards@isx.com

Abstract

Manager's associate systems enable users to indirectly manage semi-autonomous agents to support collaborative, mixed-initiative human-computer problem solving. MAX is a software framework for building manager's associate systems. It provides an architectural model, a domain-independent software structure, tools, and reusable components for building domain-specific elements of systems used by managers to develop, execute, and analyze task plans for agents. This paper presents an overview of MAX and describes its first application: a ground vehicle manager's associate system for the management of robotic vehicles exploring a simulated planetary surface.

Introduction

The WIMP user interface (Windows - Icons - Menus - Pointing Device) was introduced by Xerox's Alto and Star and popularized by the Apple Macintosh almost 15 years ago. These products were followed by several generations of new personal computer products, each improving hardware and software functionality but not extending the standard WIMP direct manipulation interface concepts. Recently, a number of papers have appeared suggesting that a new generation of user interfaces is coming that will feature semi-autonomous agents that perform intelligent functions for the user.[1, 4, 6, 7, 9, 10, 12] Alan Kay, for example, distinguishes between manipulation interfaces and management interfaces and suggests that the interface of the future will enable users to indirectly manage agents, not directly manipulate objects.[6] Similarly, Dave Smith argues that we need delegation interfaces which support delegation of tasks to the computer, not more manipulation interfaces.[12]

The rationale for agents and agent-management interfaces is that there are many tasks for which direct manipulation interfaces do not work. Such tasks are often tedious, they may require too much of the human's time, and they usually require processes better performed by the computer. Examples include searching news and email files for information of interest to a human reader,[6] developing tactics plans for an overloaded fighter pilot,[3] and exploring a planetary surface through robotic vehicles.[5] For tasks that can and should be performed by the computer while the human is doing something the computer cannot do, agents will be needed and an interface through which the human can manage the agents will be required. We call the agent management interface the manager's associate.

The manager's associate is an extension of a concept of human-computer interaction derived from several years of research and development in a variety of domains, beginning with advanced pilot aiding in the Pilot's Associate system.[2, 3, 8] An associate system enables collaborative, mixed-initiative human-agent problem solving in application domains in which the human is unable to cope with the scale or complexity of the problem solving situation. In such domains human information processing can be overloaded by very large problem spaces, too many simultaneous activities, and too much data, but full automation is not possible because the task requires human perception, judgment, or expertise. The associate system employs models of the task and the user to provide advanced user support, including workload management, error recognition and correction, adaptive aiding, context- and user-adaptive display management, and selective task automation.[2]

This paper describes a system called MAX, a software framework for building manager's associate systems. It provides an overview of the system and then describes its application in one domain: supervisory management of robotic vehicles.

The MAX Framework

Our goal in developing MAX was to leverage the lessons learned, the experience, and the architectures developed in the Pilot's Associate program to build a generic system supporting cost-effective development of manager's associate systems. This generic system must support development of applications in a range of application domains, including but not limited to avionics.

MAX is a software framework that provides an architectural model, a domain-independent software structure, tools, and reusable components for building domain-specific elements of systems used by managers to develop, execute, and analyze task plans for semi-autonomous agents. These elements can be employed to support problem assessment, focusing the user's attention on problems of interest, developing an effective problem-solving strategy, and executing the problem-solving strategy through selective task automation and human performance assistance.

Manager's associate applications are defined in MAX as a collection of Activities, which are major elements of the manager's task such as Planning, Execution, Analysis and, for control of semi-autonomous vehicles, Tele-Operation. These activities have subactivities, e.g., the Planning activity can be composed of subactivities like Assign Objectives to Resources, Plan Tasks, and Calculate Performance Measures. Each of these subactivities can in turn be composed of lower-level subactivities. For each activity, MAX provides an Activity Manager, a Data Process Manager, an Interface Manager, and a Command Monitor. The Activity Manager provides control and coordination of system resources, while the Interface Manager provides managed information to the user through a graphic user interface and provides mechanisms for the user to interact with the system. Monitors maintain and process state information of activity-specific objects and domain specific data items in a global data store. Monitors also signal alerts which may be acted upon by the associate system or the human manager, as appropriate.

A primary characteristic of manager's associate applications is that the human needs to selectively interact with large collections of data. These interactions typically include:

- Observing data, supported in MAX by modeling the classes of activities the user is involved in, locating data relevant to those activities, and presenting activity-specific abstractions of that data.

- Monitoring data, supported in MAX by data monitoring functions that apply rule-based criteria to identify the state of the data of interest to current user activities and to signal alerts based on this state.

- Selecting and applying data processing operations. For any user activity, MAX data handlers can be defined to select and apply specific operations to selected data, and can specialize the selection of operations based on the state of the data.

- Responding to alerts, supported by a mixed initiative task management scheme. For any alert condition, MAX can identify candidate responses. These responses may include:
  - Applying a data processing operation
  - Cueing a new user activity and its supporting MAX functions
  - Performing specific computations to generate additional options
  - Ignoring the alert condition

MAX decides how to handle an alert condition by applying rule-based decision models to evaluate the utility of responding to the alert, the legality of each known option, the utility of each legal option, and the utility of automating the function for the user. This process allows MAX to consider a number of factors in deciding what options to present to the user, whether any should be suggested as the "best" action for the user, or whether any options should be automatically pursued without user intervention. The factors that drive this mixed-initiative reasoning include the priority of the problems the user is facing, the user's workload, the user's preferences for selecting specific options, the user's preferences for various levels of automation, and the utility of applying specific options to the problem at hand.

The MAX framework provides a set of supporting software to enable and simplify

the construction of domain-specific applications that exhibit these capabilities. In its current form, the MAX framework provides this support by defining three categories of application developer tools:

- A set of "standard" components (and tools to define them) that can be used to construct MAX application components and specify their run-time interactions in a high level representation. These include "hooks" and tools for specifying domain-specific interface components, data definitions, and processing procedures, and "hooks" to specify user tailoring of the application's behavior.
- A collection of procedural attachments to those components that implement high level specifications of control interactions as low-level run time control decisions.
- A collection of "default" interface components.

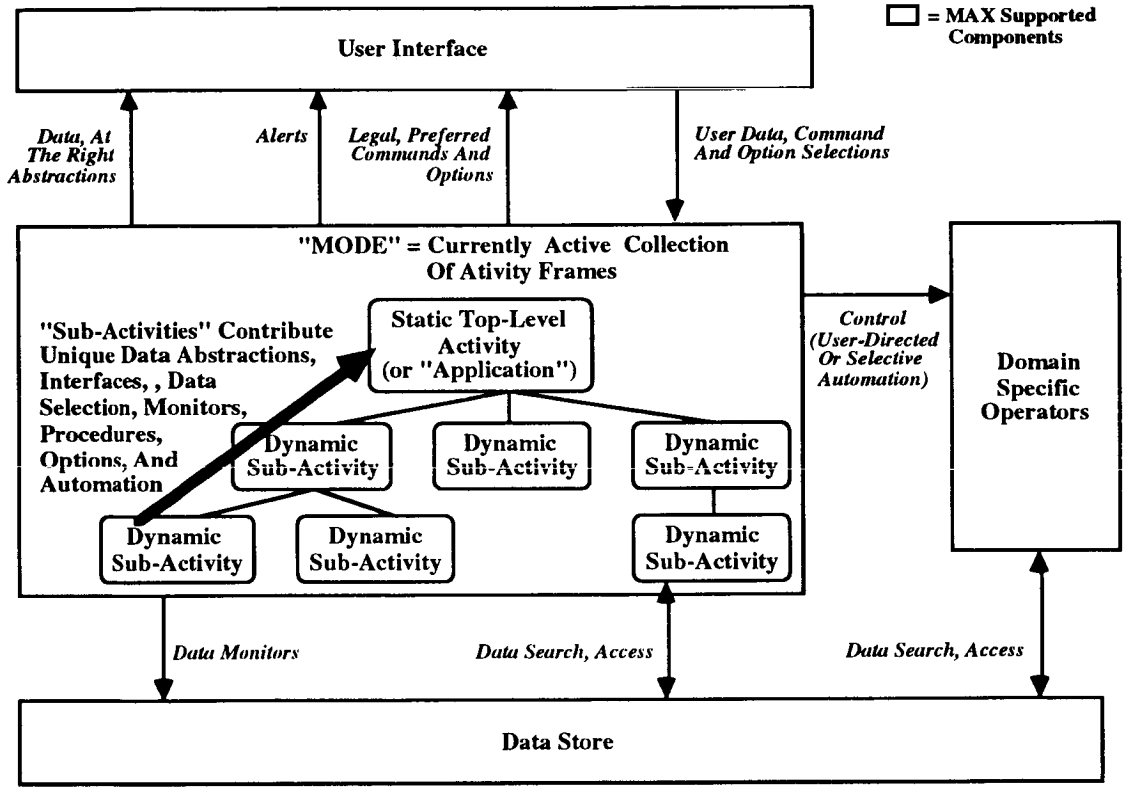

Figure 1. Run-Time Anatomy of a MAX Application

System Architecture

MAX is centered on the concept of Activities. The application user, at any point in time, is operating in a mode comprised of a currently active collection of activities. Each activity contributes interfaces, specialized tools, data of interest, and data monitors to the user's environment while operating in that mode. Figure 1 illustrates the interaction among activities, composing and maintaining a unique combination of sub-activities at run time to respond to both the dynamically changing set of problems confronting the user and the user's own direction for reacting to those conditions.

Each activity in MAX is composed of a few basic components, repeated as needed to define a complete activity. These components, along with their primary interactions, are shown in Figure 2. The top level MAX activity, upon activation, creates its interface manager and data manager. The interface manager in turn creates appropriate

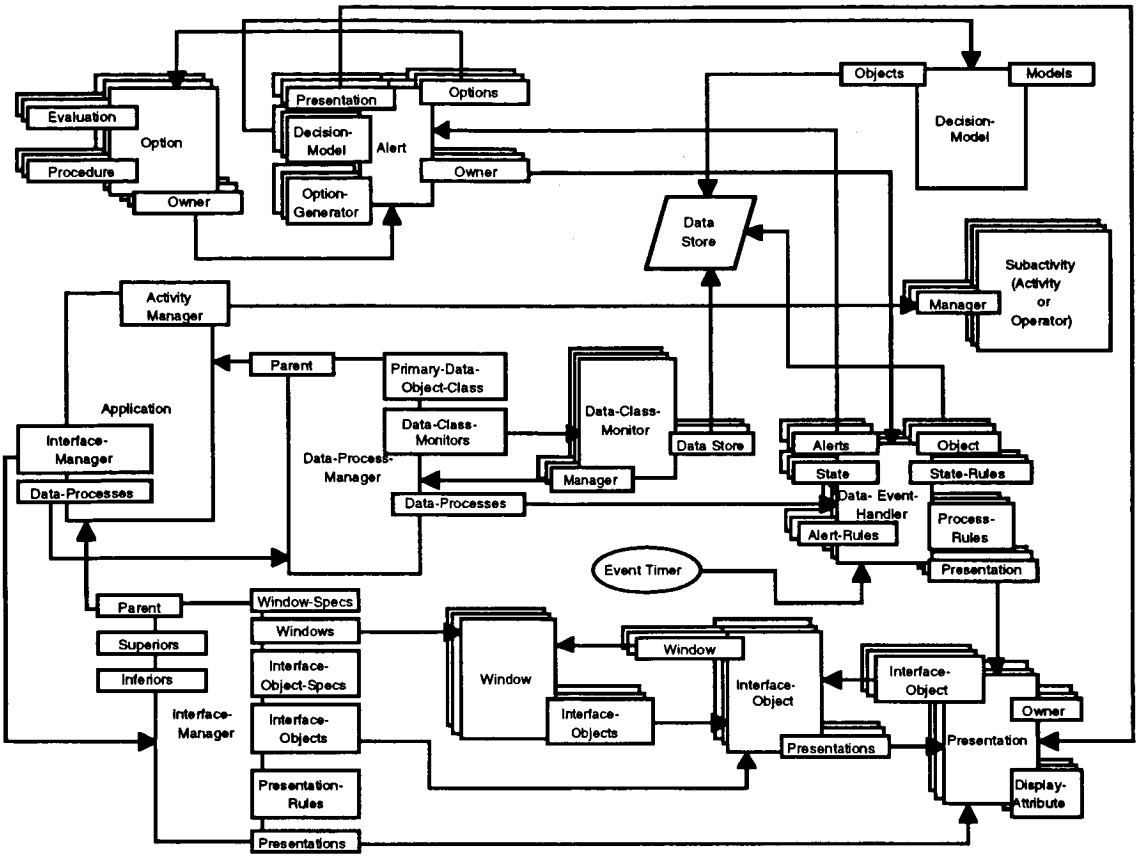

Figure 2. Anatomy of a MAX Activity Frame

instances of MAX's built-in interface objects (windows, menus, buttons, presentations, etc.) along with those domain-specific interface components specified in the activity definition.

The activity also creates a data manager, which establishes data class monitors to detect objects of interest in the data store. Each object of interest found by these monitors (either immediately or at any time in the life of this activity) causes a data-event-handler to be created, and a data-instance-monitor to be established. This instance monitor detects any changes in the object of interest, and triggers data change events, These in turn cause the data-event-handler to reevaluate the object of interest, and may cause it to either trigger associated data processing operators, or may trigger an alert condition. For data whose impact on the problem may change over time rather than over changes in value, a clock-driven event timer is provided to re-trigger data evaluation.

In an alert condition, an alert is asserted. The alert in turn generates a set of possible options to deal with that alert. Options consist of three classes of operations: data processing procedures; procedures that generate new options; and invocation of sub-activities to help the user solve the problem. Next, the alert determines which options are legal in the current context, and then evaluates the utility of applying each legal option. For the best options, the alert also evaluates the utility of automating the option's execution, as opposed to presenting it for the user's consideration. In each of these steps, "evaluating utility" employs rule-based decision models that consider problem characteristics, the user's current problem solving workload, the user's personal preferences among options and for his desired level of automation support, and models of the capabilities and current state of the MAX application.

Implementation

MAX was developed in Macintosh™ Common Lisp, version 2.01p3 and Expertelligence's Action!™, version 3.0. It runs on a Macintosh IICI with 20MB of RAM, an 80MB hard drive, and a 13 " or larger color or gray scale monitor. The system can run in other Macintosh environments, including a Powerbook™, with modification of the map displays.

The GVMA

MAX is designed to support a wide range of manager's associate applications. The choice of the Ground Vehicle Manager's Associate (GVMA) was made in collaboration with NASA Ames to support a successful demonstration of simulated planetary exploration using IS Robotics, Inc. robotic vehicles.*

The GVMA has three main activities, the mission planning activity, the mission management activity, and the vehicle tele-operation and video survey activity. A robot simulation activity was also created to provide an internal vehicle simulation that can be executed from within the GVMA.

Monitoring the activities of all vehicles, the system alerts the manager to events requiring human attention, provides options for human action, and delivers vehicle commands that implement options invoked by the manager. The manager monitors the sensors and can directly control each vehicle through the Tele-Operate activity window, which includes a live video feed from the vehicle. The overall activity of the vehicles is displayed in the Mission Management window. The GVMA observes the performance of each activity against a mission plan. When a situation triggers a monitor, the GVMA displays an alert and suggests actions to the manager through an Alert window. If the manager invokes one of the suggested options, the GVMA executes the selected action. Figure 3 shows an example in which an alert has signaled readiness to perform a search activity at a location on the planetary surface. At this point, the manager can tell the system to invoke the search immediately, wait for the start time specified in the mission plan, or wait for further direction. If the activity is invoked, the GVMA will cause the vehicle to proceed to the search area via the specified waypoints.

Assessment

The GVMA demonstration showed that MAX provides a framework for building an associate system for managers of multiple vehicle missions. We plan to apply MAX to the development of manager's associate systems in different domains in the near future.

Using MAX, the GVMA was developed in two person-months, including interfacing to the robot vehicles. We did not attempt to build a non-MAX GVMA in order to compare development time and difficulty, but based on our experience we believe that an application of the complexity of the GVMA would have required at least six person-months if developed from scratch.

Although this suggests that MAX has value in developing manager's associate applications, it is not a finished product. We think that it was helpful in developing the GVMA, but enhancements are required. Facilities for configuring communications interfaces and for developing task, environment, and user models work, but they are difficult to use. In addition, the software is not as robust as it should be. MAX supported the development of the GVMA, but it has not been tested in a variety of applications, and such testing will almost certainly reveal undetected bugs. Finally, more effort needs to be invested in developing planning and decision analysis functionality. The decision analysis module works, but it is fairly primitive and decision models are difficult to build and revise. The current planning module requires the user to develop plans using a very simple editor, and there is no replanning capability. Determining the effectiveness of the GVMA and other manager's associates is problematic. As noted by Sheridan,[11] the objective function of supervisory control is not fixed and cost and complexity make it extremely difficult to conduct controlled experiments in this domain. It is certainly possible to demonstrate the superiority of the GVMA over manual control, but

*IS Robotics, Inc., Twin City Office Center, Suite 6, 22 McGrath Highway, Somerville, MA 02143.

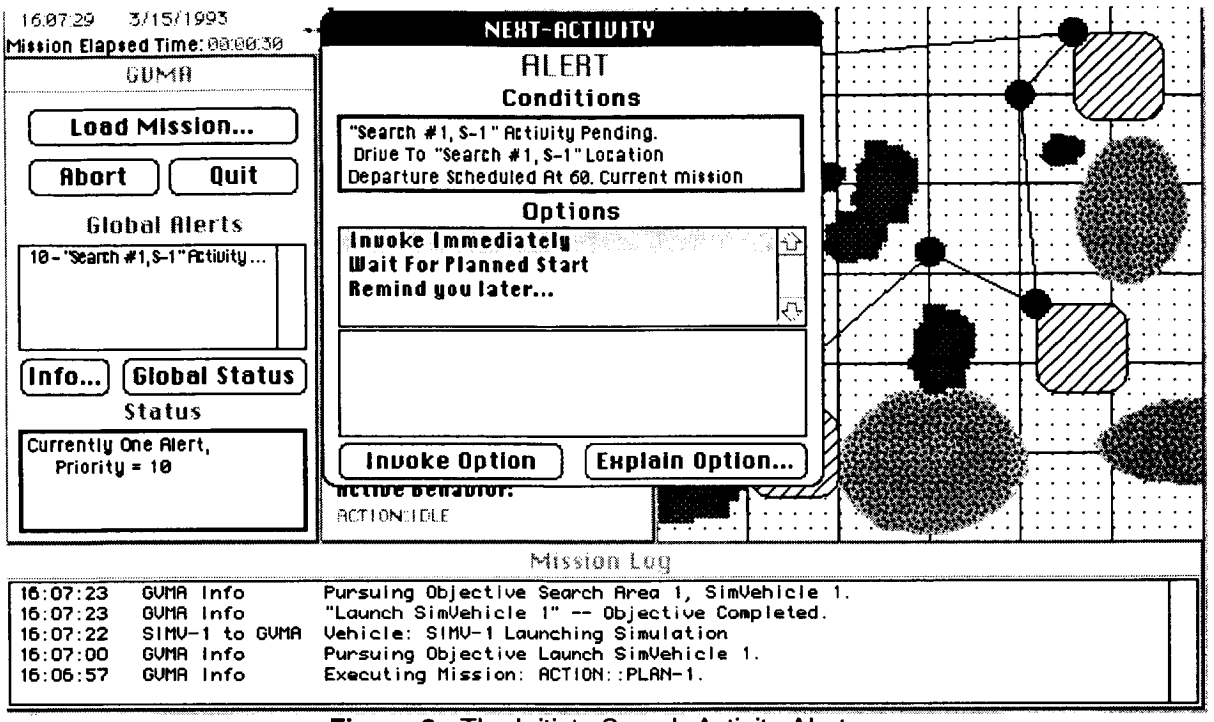

Figure 3. The Initiate Search Activity Alert

understanding the relative effectiveness of the many forms and combinations of associate features and behaviors is another matter. We are investigating ethnographic and usability inspection methodologies as adjuncts to the methods of experimental psychology.

Summary and Future Work

Agents and manager's associates are required by many important new applications, and we will be seeing commercial implementations of such interfaces in the near future. The first products will probably support information access applications that employ agents in storing, retrieving, manipulating, and understanding massive amounts of information. The management of intelligent devices such as remote vehicles, manipulators, and instrumentation will be another important application. These applications will require a manager's associate, and we believe that cost-effective development will require a framework like MAX.

As noted in the discussion of effectiveness, MAX is not a finished product. We plan to continue work on the system, improving it as we use it build new applications. First steps are enhancement of the planning and decision analysis functionality. We are also working with IS Robotics, Inc. to extend the GVMA and interface it to new micro-robot platforms.

Acknowledgements

MAX and the GVMA were developed with support from NASA Ames Research Center Small Business Innovative Research contracts. IS Robotics, Inc. supplied the robotic vehicles, and Dr. Carl Friedlander was responsible for developing interfaces and on-board behavior code. Dr. David Korsmeyer was responsible for management of the GVMA demonstration at NASA Ames Research Center.

References

1. Card, S.K., Robertson, G., and Mackinlay, J.D. The Information Visualizer: An information workspace. In *Proceedings of SIGCHI '91*, 1991, 181-188.

2. Edwards, G.R. and Geddes, N.D. Deriving a domain-independent

architecture for associate systems from essential elements of associate behavior. In *Proceedings of the First DARPA Workshop on Associate Systems Technology*, 1991.

3. Fields, C.I.. & Kushner, B.G. The DARPA Strategic Computing Initiative. In *IEEE Proceedings of COMPCON*, 1986.

4. Fischer, G., Grudin, J., Lemke, A. C., McCall, R., Ostwald, J., Reeves, B. N., and Shipman, F. Supporting indirect, collaborative design with integrated knowledge-based design environments. *Human Computer Interaction*, 7, 3, 1992, 281-314.

5. Geddes, N. and Hoffman, M. Supervising unmanned roving vehicles through an intelligent interface. In *Proceedings of SOAR '91*, 1991.

6. Kay, Alan. User interface: A personal view. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1990, pp. 191-207.

7. Laurel, Brenda. Interface agents: Metaphors with character. In B. Laurel (Ed.), *The Art of Human-Computer Interface Design.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1990, pp. 355-365.

8. Lizza, C. and Friedlander, C. The Pilot's Associate: A forum for the integration of knowledge based systems and avionics. In *Proceedings of NAECON*, 1988.

9. Nielsen, Jakob. Noncommand user interfaces. *Communications of the ACM*, 36, 4, (April 1993), 83-99.

10. Robertson, G., Card, S.K., and Mackinlay, J.D. Information visualization using 3D interactive animation. *Communications of the ACM*, 36, 4, (April 1993), 57-71.

11. Sheridan, T.B. Supervisory control of remote manipulators, vehicles and dynamic processes: Experiments in command and display aiding. *Advances in Man-Machine System Research*, 1, 1984, JAI Press, 49-137.

12. Smith, David C. Common elements in today's graphical user interfaces: The good, the bad, and the ugly. In *InterCHI '93 Conference Proceedings*, Amsterdam, April 1993, 470-473.