IDIS Technical Report 101

Active User Interfaces

Scott M. Brown
Crew System Interface Division
Air Force Research Laboratory
Wright-Patterson AFB, Ohio 45433-7022

Eugene Santos, Jr.[1]
Department of Computer Science and Engineering
University of Connecticut
Storrs, Connecticut 06269

September 21, 1999

IDIS
Intelligent Distributed Information Systems

Abstract

The current state of user interfaces for large information spaces imposes an unmanageable cognitive burden upon the user. Determining how to get the right information into the right form with the right tool at the right time has become a monumental task. While advances in graphical user interfaces can partially address this problem, the basic problem of information overload–and under load–can not be solely addressed through development of a better direct manipulation interface to the information space.

We survey the state of the art in two research fields, interface agents and user modeling, that address the problem of information overload and under load. First, we take a historical look at how the fields of human-computer interaction and artificial intelligence have viewed interface agent research. Interface agents address the problem of increasing task load by serving as either an assistant or associate, extracting and analyzing relevant information, providing information abstractions of that information, and providing timely, beneficial assistance to users. Interface agents communicate with the user through the existing user interface and also adapt to user needs and behaviors. User modeling is concerned with how to represent users' knowledge and interaction within a system to adapt the system to the needs of users. The inclusion of a user model within the overall system architecture allows the system to adapt its response to the preferences, biases, expertise level, goals and needs. We provide a comprehensive look at user modeling to include the elicitation, specification and design, verification and validation, and utilization of user models.

Even though interface agents have made some progress towards addressing the data overload and under load problem, there is active debate about whether agents or direct manipulation interfaces provide better access to information and services. Our paper culminates with a discussion on active user interfaces. Active user interfaces are the next logical step towards addressing the information overload and under load problem. Whereas interface agents are designed around the assistant and/or associate metaphor, active user interfaces serve as an *actuator* to the human-machine interface. The user of an active user interface is fully aware of any actions, whether explicit (authorized consent) or implicit (implied consent), taken by the active user interface and has a complete, intuitive understanding of such actions. We provide examples of first-generation active user interfaces within the domain of an natural language query information management system and decision support system and show how the interface is capable of providing an intuitive, meaningful, and useful "actuation" interface for a heterogeneous collection of information sources and tools. The primary goal of active user interfaces is to autonomously react to changes in user intent as well as the information sources, providing a way for users to interact naturally with the information. As a direct result, the user's task load can be reduced.

Keywords: active user interface, user modeling, user intent, relevancy, intelligent user interface, intelligent agent

[1]This work was supported in part by AFOSR Grants #940006, #9600989, and F49620-99-1-0059.

1 Introduction

Nearly every software application on the market today has some sort of user interface. As these applications become more and more complicated and complex, more is demanded of the user to perform his/her tasks. Furthermore, the voluminous amounts of information users must process increases the users' task load. On the other end of the spectrum, the inability to find the right information for the task at hand causes an information under load. Determining how to get the right information into the right form with the right tool at the right time has become a monumental task. The need exists for solutions to address the problem of increasing task load that is overwhelming the human user. These software solutions could help alleviate user task load by providing abstractions and intelligent assistance that communicates with the user through the existing user interface and adapts its assistance to dynamic user needs and behaviors. To address the information under load problem, the user must be actively engaged in a "dialogue" with the interface. To provide this dialogue, these software solutions must proactively search for and process relevant information based on the user's needs, goals, tasks, and preferences and present it to the user in a meaningful way. The solution should be robust insofar as its benefits are generic and can be applied to any highly interactive and information intensive software system[2]. Examples range from virtual battlefield management systems to freight and parcel management systems to Wall Street financial investment and analysis.

Reducing user task load involves providing intelligent assistance to the user. Providing intelligent assistance and performing tasks on the user's behalf requires an understanding of the goals the user is performing, the motivation for pursuing those goals, and the actions that can be taken to achieve those goals. The term *user intent* denotes the actions a user intends to perform in pursuit of his/her goal(s). The term *user intent ascription* is the attribution of actions to the goal(s) a user will pursue. That is, user intent ascription is the process of determining which actions are attributable to a specific goal or goals.

To ascribe user intent, system designers must identify the salient characteristics of a domain environment and specifically determine goals a user is trying to achieve, the reason and/or cause for pursuing those goals, and the actions to achieve those goals [26]. This approach is based on the belief that what a user intends to do in an environment is the result of environmental events and/or stimuli occurring in the environment and by the goals they are trying to obtain as a reaction to the events and stimuli. That is, the reason *why* users perform actions is to achieve goals they pursue as a result of environmental stimuli. Therefore, for a system to be able to assist the user in pursuing those goals, the system must be capable of ascribing user intent to offer timely, beneficial assistance. An accurate user model is considered necessary for effective ascription of user intent.

During the 1980s and 1990s, interface agents and user modeling have become a popular way[3] of helping users deal with the complexity of current environments. The purpose of these *interface* or "personal assistant" agents is to solve the problem of information overload and under load by collaborating with the user, performing tasks on the users' behalf [105]. Examples of interface agents include office assistance agents, such as e-mail, scheduling, and financial portfolio management agents [105, 157, 18]; tutor and coach agents [34, 38]; and character-based assistants for word processors, spreadsheets, and presentation software, such as the Office Assistants found in Microsoft's Office '97 software [78, 80].

[2]We use the term "system" in the broadest sense.

[3]The attendance at the Autonomous Agents conferences is a testament to this observation.

It is widely agreed interface agents' "decisions" must be based on an accurate *user model* of the users' knowledge and interactions with the system to effectively and efficiently predict user intent. User modeling is concerned with how to represent the user's knowledge and interaction within a system to adapt the system to the needs of the user. Researchers from the fields of artificial intelligence, human-computer interaction, psychology, education, as well as others have investigated ways to construct, maintain, and exploit user models. The benefit of utilizing a dynamic user model within a system is to allow that system to adapt over time to a specific user's preferences, work flow, goals, disabilities, etc. To realize this benefit, the user model must effectively represent the user's knowledge and intent within the system to accurately predict how to adapt the system. The elicitation, specification, design, and maintenance of an accurate cognitive user model is necessary for effective ascription of user intent.

User models are particularly useful in domains with a heterogeneous group of users and where the system exhibits some flexibility in its "response" to the users [139]. For example, an automatic teller machine (ATM) is definitely used by a heterogeneous group of users, but the "response" it may give to users' interactions with it are limited and not, per se, a function of users' knowledge. On the other hand, a World Wide Web search engine or database interface is a perfect domain for user models. To be sure, we are seeing a plethora of user models being used in these domains [131, 1, 5, 105].

The investigation of tools, techniques, methodologies, and theories for addressing the problem of increasing user task load as the result of information overload and under load is the main goal of this paper. This paper is broadly segmented into three areas: interface agents, user modeling, and active user interfaces. In particular, we survey the state of the art in interface agents and user modeling as these two research fields relate to addressing the problem of information overload and under load. Section 2 discusses interface agents and how the fields of artificial intelligence and human-computer interaction have largely influenced the research conducted by interface agent researchers. In Section 3, we provide a historical background of the emergence of user modeling as a research discipline, as well as introduce the concept of user intent and how user models may possibly be used for ascription of user intent within a domain. This section also details the elicitation (i.e., acquisition) of user models. and a critical discussion of the specification and design of user models from the viewpoint of what is relevant to determine user intent.

Even though interface agents have made some progress towards addressing the data overload and under load problem, there is active debate about whether agents or direct manipulation interfaces provide better access to information and services [107]. Our discussion culminates with a discussion on active user interfaces. Active user interfaces provide a new user interface metaphor and are the next logical (and evolutionary) step towards addressing the information overload and under load problem. Whereas "interface agents" are designed around the assistant and/or associate metaphor, active user interfaces serve as an *actuator* to the human-machine interface. We discuss the key characteristics of active user interfaces (e.g., adaptive, adaptable, analytic, autonomous) These interfaces go significantly beyond the adaptable and adaptive capabilities of existing interfaces. These interfaces are capable of multi-levels of collaboration and autonomy. The user of an active user interface is fully aware of any actions, whether explicit (authorized consent) or implicit (implied consent), taken by the active user interface and has a complete, intuitive understanding of such actions. We provide examples of first-generation active user interfaces within the domain of information management and decision support systems and discuss how such an interface is capable of providing an intuitive, meaningful, and useful "actuation" interface for a heterogeneous collection of information sources and tools. The active user interface must maintain a dynamic user model of the relevant concepts in the user inquiries as they relate to the information sources and tools. The primary goal of active user interface research within these domains is to autonomously react to changes in user intent as well as the information sources and tools, by dynamically constructing the appropriate queries relative to the changes identified and providing a natural way for users to interact with these information sources and tools.

2 Interface Agents

While the term "expert system" was the artificial intelligence buzzword of the 1980s, "agent" is the 1990s' buzzword. [167], [57], and [131], as well as others, have surveyed the plethora of agent definitions. While there are many types of agents (cf. [57]'s agent taxonomy), this paper is concerned only with interface agents. For concreteness, the usage of the term "interface agent" is defined, using a hybrid of several definitions found in the literature:

> An adaptive interface agent is a software system situated within a domain environment that adaptively senses, acts, and reacts within this environment over time, to pursue its own goals and the goal of providing collaborative assistance[4].

Interface agents are a relatively new area of research within the human-computer interface (HCI) and artificial intelligence (AI) communities[5]. The use of interface agents within a system usually take the form of a personal assistant [105] or associate metaphor [7, 71, 116]. These agents are *active* and *cooperative*, possessing the ability to adapt their "behavior" to the needs of the user based on past and present states of affairs [30]. "Active interfaces" are pro-active, autonomous, and adaptable [32]. Their cooperative nature is the result of the user-defined *task delegation* to control their performance of tasks for the user [31].

Not all applications are suited for the addition of an interface agent. [165] states,

> The main application areas for intelligent interfaces are thus such where the knowledge about how to solve a task partially resides with the computer system. Since the user does not know exactly what should be done, he or she cannot manipulate the computer as a tool, but must ask the system to do something for him or her. This request may be incomplete, vague or even incorrect given the user's real needs.

[165] requires intelligent interfaces to exhibit the following techniques to connote "intelligence" in interfaces:

- **User Adaptivity**—Techniques that allow the user-system interaction to be adapted to different users and usage situations.
- **User Modeling**—Techniques that allow a system to maintain knowledge about a user.
- **Natural Language Technology**—Techniques that allow a system to interpret or generate natural language utterances, in text or in speech.
- **Dialogue Modeling**—Techniques that allow a system to maintain a natural language dialogue with a user, possibly in conjunction with other interaction means (multi-modal dialogue).
- **Explanation Generation**—Techniques that allow a system to explain its results to a user.

[4]Assistance is broadly defined to mean timely, beneficial suggestions, tutoring, help, interface adaptations, etc. for the user to the user.

[5][85] add object-oriented programming and concurrent object-based systems disciplines. We consider these disciplines to be more paradigm-oriented (i.e. the object oriented paradigm), where agent-oriented programming [150] is yet another programming paradigm. To be sure, [85] state "object oriented programmers often fail to see anything novel or new in the idea of agents."

[77] outlines a number of problems that must yet be solved by these research communities before intelligent user interfaces become a reality. In particular, she states there is a need to develop:

- Usability principles for *intelligent* interfaces versus direct-manipulation systems;
- Reliable and cost-effective intelligent user interface development methods;
- A better understanding of how intelligence can *improve* the interaction;
- Authoring tools that enable easy development and maintenance.

[30] add "it will be necessary to develop theories underlying these systems, as well as to include experimentation phases after having defined suitable metrics."

The next two subsections discuss the relationship of interface agents to both the HCI and AI communities in terms of the development history, how both approach the field, and their strengths and weaknesses[6]. While there is no clear delineation between the two communities with regards to their research in interface agents, both approach the research field of interface agents differently. Section 3 ends with a discussion on how the field of user modeling has brought the HCI and AI communities together for the purpose representing users' knowledge and interaction within a system to adapt those systems to the users' needs.

2.1 HCI Historical Development

One word that characterizes the human-computer interface (HCI) community's research into interface agents is "customization." The HCI community has concerned itself with what the user can do *with* the agent and the interaction between human and agent. HCI researchers concern themselves with how techniques and methodologies affect *user* performance. The strength of HCI research in interface agents is its attentiveness to the user, focusing on what a user needs to perform his/her tasks, and how best to represent information to the user.

[70][7] presents 10 research questions of user/expert system interactions, focusing on control and communication of relevant information between the user and system:

1. How do people think about tasks they perform and what trade-off strategies are used in performing them?
2. How can human tasks be translated into conceptual/cognitive abilities needed to perform a task generally/specifically?
3. Given 2), how can they be codified in a computer?
4. How does a computer inform a user what task it is performing?
5. When is a task appropriate to perform; when/how should the user be informed (voice, text, graphics, other)?
6. How would an intelligent/complex program display its thinking/chain of reasoning in a way a user can easily understand?

[6]While we do not mean to stereotype a particular research field too narrowly, it is advantageous to point out the general "slant" different research communities take on interface agent research.

[7]As referenced in [140].

7. What level of sophistication and filtering is necessary and sufficient to display the computation going on?
8. How rigorously can graphic information or other symbolism serve as a filter for deeper (more detailed) knowledge?
9. What information should be thrown away (not displayed) as irrelevant, ambiguous, or inappropriate to a given situation?
10. What kind(s) of integrated workstations are appropriate for different applications?

[5] present a history of HCI research[8]. They trace the research from customizable systems to intelligent (interface) agents. Three approaches frame the direction of HCI research:

1. Educate developers about users and their work.
2. Form active collaborations among users and developers when designing systems.
3. Shift responsibility for final design decisions from the initial developers to the users or to people who are closer to the users.

The last approach has been central to the agent research performed by the HCI community. *Customizable* systems allow end users to adjust a system to their specific needs and tasks. These adjustments can be as simple as allowing a user to choose from predetermined alternatives to providing users a way to alter the system itself. Related to this customization is the ability for users to share useful customization with other users.

Several user customization techniques exist, including preferences and templates, customizable workspaces (e.g., modifiable tool bars), end-user programming scripts (e.g., mail filtering rules), and programming-by-example (e.g., any macro recorder). Adaptive interfaces are another customization technique, but one where the user is not in complete control. Adaptations are done either via statistical averages of users or dynamic user models [5]. Examples of the use of statistical averages include the adaptation of hyper-media menus based on frequency of selection [1] and the method used by [75] to select tools, communication modes, and files. Intelligent tutoring and coaching systems are an example of systems with dynamic user models [38].

The 1990s saw the emergence of "intelligent" interface agents. Within the realm of HCI research, interface agents are considered "personal assistants" collaborating with the user. Several design considerations are typically considered by the HCI community and include the following [5]:

- Give the user a feeling of control.
- Pay attention to the nature of the human-agent interaction.
- Employ built-in safeguards (protect the user).
- Provide the user with accurate expectations.
- Cater to privacy concerns.

[8][121] summarizes the (brief) history of HCI technology from the perspective of the impacts of university research on key technologies. Of note, he does not mention interface agents until his final paragraph.

- Hide complexity.

HCI has also been concerned with how to represent agents to the human [91, 16]. Related to this representation are the issues of trust, anthropomorphization, and privacy. Many researchers are investigating ways to anthropomorphize agents in an attempt to make the agents more personable and increase trust in the agent's abilities [94, 51, 86, 4]. HCI researchers are concerned with the ways to convey the agent's internal state to a user. If the old adage "first impressions are lasting impressions" is taken to heart, HCI researchers are concerned with the user's impressions with the agent. If a user does not trust the agent—typically as a result of failing to understand the agent's "abilities"—future interaction will be affected. [91] investigate the affect of anthropomorphizing (via adding "faces") the interface. [16] investigate the use of a rule-based expert system together with voice, gesture, body position and context in a virtual reality interface.

2.2 AI Historical Development

A word that characterizes the AI community's research into interface agents is "delegation." In contrast to the HCI community, the AI community as a whole has concerned itself with what an interface agent can do *for* the user. AI researchers concern themselves with how techniques and methodologies affect *agent* performance. The strength of AI research in interface agents is its attentiveness to knowledge representations for capturing the "state of the world" and how to adapt the agent's behavior using machine learning techniques. The community is also concerned with models of agent communication.

[104] states that agent research began in 1985, when the term "animat approach" was coined. [104] also does an excellent job of distinguishing "traditional AI" from the study of autonomous agents (what we'll term *agent AI*). Table 1 summarizes the main differences traditional AI and agent AI.

[104] discusses the basic problems AI researchers have with adaptive autonomous agents, which includes interface agents. The two basic problems are the following:

- **Action selection**—What to do next given time-varying goals?
- **Learning from experience**—How to improve performance over time?

These problems are far from solved and attempts to address them are active research topics.

In comparison with HCI research for allowing users to explicitly share useful customizations with one another, AI techniques have revolved around agents' communication with one another to share useful information and theories modeling that communication. Architectures and models have been built around agent cooperation and negotiation [171, 119]. The term "agency" has evolved out of this area of research. Several AI interface agent projects have successfully married the approach of sharing users' customizations with agent communication [105].

Whereas the HCI approach to interface agent research focuses on a collaboration between human and agent, AI research has relied heavily on the strength of other AI research fields, in particular knowledge representation, machine learning and planning. For example, compare the collaborative nature used by the Agent Building Shell for completing conversation rules [10] with [105]' approach to learning new situation-action pairs [105]. The former relies on the user to help specify incomplete rules while the latter relies on case-based reasoning to "recognize" new situations. AI concerns itself with how techniques and methodologies affect *agent* performance.

Table 1: Features of Traditional AI and Agent AI

TraditionalAI	*AgentAI*
Focuses on "depth" of knowledge and competence within a domain	Focuses on "breadth" of knowledge and exhibits a wide diversity of competence, but at a shallower level
Focuses on "closed" systems	Focuses on agents situated *in* the environment, which is typically dynamic in nature
Contends with one problem at a time	Contend with multiple, competing problems
Knowledge is static because it models domain expertise	Knowledge is dynamic because it models users
Not concerned with the developmental aspects of the knowledge structure	Relies on adaptivity of the knowledge structures, incrementally modifying the structures

2.3 Interface Agent Examples

In recent years, a plethora of interface agents have emerged. This section is by no means an exhaustive enumeration of interface agents. The interface agents presented were chosen for several reasons. Some were chosen for their historical significance (e.g., they were the first of their kind), while others were chosen for the contribution they made to the field of interface agent research, while yet others were chosen because they represent the "cutting edge" of current interface agent research. This section provides a short background on the types of interface agents and the domains where they have been used to date. [98] segregates agents for human-agent interaction (HAI) into three categories: anticipatory, filtering, and semi-autonomous.

Anticipatory agents attempt to automate some portion of the action-execution cycle. The purpose of these agents is to use information gathering techniques within the confines of the application domain to determine a context used to infer the user's intent. Many of these agents are constructed as adaptive agents, learning a user's preferences, abilities, goals, and needs over time. Two subclasses of anticipatory agents exist: (1) those attempting to learn a user's categorization scheme and (2) those that seek to infer a user's plan of action by observing atomic actions within the domain.

A number of anticipatory interface agents have been integrated into complex application environments. CAP (Calendar APprentice) [117] is an agent that learns room scheduling preferences. Users enter and edit meetings on a calendar and instruct the CAP agent to send invitation to the invitees. CAP uses the "looking over the shoulder" approach [106], acquiring rules for predicting the duration, location, and times of meetings. [106] and [95] present a similar scheduling agent using situation-action pairs to recognize the context of a user's interactions and the appropriate action to take given the situation. COACH is an advisory system for users writing Lisp programs [148].

COACH builds an adaptive user model of a user's experience and proficiency, and using its domain knowledge of Lisp selects appropriate and timely advice (e.g., a function definition, example, syntax, style guide, etc.) to offer the user. Lumiére is a multi-year research project for integrating intelligent assistance into the Microsoft Office product line [80]. Lumiére uses a Bayesian network user model to determine the probable goals a user is pursuing and the needs associated with the goals to provide assistance to users in meeting those needs. The approach used in Lumiére is also used in an intelligent assistant called LookOut for Outlook's calendaring sub-system [79]. Remembrance Agent [138] is an automated information retrieval system agent, integrated into the Emacs environment [151]. Remembrance Agent uses memory-based reasoning [152] to help users remember relevant information related to their current goal. Physician's Assistant [11], based on IBM's Ginkgo technology [81], also uses memory-based reasoning. The assistant helps doctors prescribe drugs, including frequency, duration of treatment, and whether it should be taken with meals, for a given situation and recalls a doctor's own practice pattern.

Filtering agents are similar to anticipatory agents in that they also attempt to automate some user actions. Specifically, the filtering agents are concerned with how to automate a user's "filtering mechanism" for determining what is relevant given a scenario and only presenting the relevant information, filtering out the irrelevant information. Examples of filtering agents include the following:

- Maxims [96], an e-mail filtering agent;
- Ringo [105], a music recommendation system;
- Letizia [100, 101], one of the first World Wide Web (web for short) "surfing" agents utilizing simple keyword-frequency information retrieval to suggest relevant "near-by" web pages;
- CiteSeer [17, 97], a web agent for automatic retrieval and identification of relevant publications;
- WebMate [33], a personal web information retrieval agent utilizing multiple heuristics for determining relevance for further browsing and searching;
- WebAce [73], a web agent that automatically categorizes and filters a set of documents and performs new queries used to search for relevant related information;
- SHOPPER'S EYE [53], a personal digital assistant (PDA)-based agent that uses the concept of physical location-based filtering via global positioning satellites (GPS) to support mall shopping.
- PHOAKS (People Helping One Another Know Stuff) [158], a collaborative Usenet newsgroups filtering system that "mines" recommended Web pages from over 6,000 newsgroups.

Semi-autonomous agents are characterized by a user's explicit activation of the agents to perform tasks on the user's behalf. Activation of the agents occurs via one or more instructional interfaces: form-based interfaces, high-level scripting languages, direct-manipulation interfaces, programming by example, and visual languages.

One of the earliest semi-autonomous agents was Information Lens [109]. The agent's domain is an electronic mail application. Information Lens assists users with the filtering, sorting, prioritization, and general management of e-mail. Users instruct Information Lens via e-mail composed as semistructured messages. The messages are user-defined templates that specify message type (e.g., announcement) with additional header fields (e.g., topic, date). Users import, create, or modify existing active rules for processing e-mail. These active rules serve as simple semi-autonomous agents, managing a user's e-mail on their behalf.

One of the most well known software agents employing the semi-autonomous notion is the Internet Softbot [52]. Softbot uses a Unix shell and the World Wide Web to interact with a wide range of Internet resources. For example, the softbot can be "told" to locate all papers that reference this paper. The softbot provides an integrated and expressive X-windows, form-based graphical user interface to the Internet, but one which the user explicitly activates. The interface shields users from the underlying subset of first-order logic used as a goal planning language. As a "personal assistant," the softbot is robust in that ambiguity, omission, and errors in user requests are tolerated.

Programming-by-example systems create generalized programs from examples provided by users [47] and bridge the gap between anticipatory and semi-autonomous agent systems. Eager is a programming-by-example system for the HyperCard environment [47]. It monitors the user's activities and when it detects an iterative pattern, it writes a program to complete the iteration. When Eager detects a repetitive activity, it highlights menus and objects on the screen to indicate what it expects the user to do next. When the user has developed confidence and trust in Eager's suggestions, he/she can instruct Eager to complete the tasks for him/her. Mondrian, an object-oriented graphical editor that can learn new graphical procedures, is another system utilizing programming by example [99]. A user demonstrates a sequence of graphical editing commands on a concrete example to illustrate how the new procedure should work. An interface agent records the steps of the procedure in a symbolic form, using machine learning techniques, tracking relationships between graphical objects and dependencies among the interface operations. The agent generalizes a program that can then be used on "analogous" examples. The generalization heuristics set it apart from conventional macros that can only repeat an exact sequence of steps. The system represents all operations using pictorial "storyboards" of examples.

3 User Modeling

User modeling, as a research discipline, emerged in the 1980s, from research being done in the artificial intelligence, human-computer interaction, psychology, education, and other research fields. In the previous sections, we concentrated on the first two research fields—HCI and AI—discussing their impact on the development of interface agents and their strengths and weaknesses. In this section, we discuss how the field of user modeling has brought the HCI and AI research communities together for the purpose of representing users' knowledge and interaction within a system and a discussion on user intent and its applicability to user modeling. Several of the ideas presented in this section, as well as the general flow of the presentation, follows Loren Terveen's article [159] "Overview of Human-Computer Collaboration."

3.1 User Modeling Historical Development

The AI community is concerned with the internal knowledge representation of interface agents versus the HCI community's concern with representation of the agent to the user. The field of user modeling is perhaps a good marriage between artificial intelligence and human-computer interaction. User modeling is concerned with how to represent the user's knowledge and interaction within a system to adapt those systems to the needs of users. [14] state that user models are one of three models[9] all adaptive systems must possess. The main purpose of user modeling is to determine what the user intends to do within a system's environment for the purpose of assisting the user. [125] defines human (user) intent as "mental states which drive actions." As previously stated, the term "user intent" is used to denote the actions a user intends to perform in pursuit of his/her goal.

The infusion of ideas from many disparate research fields has allowed user modeling researchers to benefit from the important contributions of each of the separate contributing research fields. For example, the user modeling community has been able to reap the benefits provided by artificial intelligence researchers by various knowledge representations developed by AI researchers. The methods used include logic-based techniques [133, 61], abductive reasoning-based techniques [46, 45], machine learning techniques [35, 50, 127, 65, 3], Bayesian methods [38, 75, 22, 2, 123, 42, 80], and neural networks [127, 65, 3]. The human-computer interaction research impact on user modeling can be seen in the use of user models to customize presentation of information [67, 76, 87], to provide feedback to users about their knowledge in a domain [29], and to help users locate useful information [103, 108, 1]. Additionally, user models have taken into account various human factors, such as a user's psychological ability, working memory, task load or cognitive load, to adapt a user interface and/or the information presented to the user [118, 147, 154, 59, 8, 24, 90, 15]. The use of user models has been shown to increase effectiveness and/or usability of systems implementing the various techniques from the field of user modeling [83, 69].

[162] believes researchers need to start using lessons learned from user modeling to impact the way they view interactive human-computer environments. She discusses viewing these environments along three orthogonal dimensions: elements—the goals, plans, resources, and actions composing the atomic entities an agent (human or otherwise) is concerned with; processes—the types of processing (e.g., reaction, deciding, learning) that takes place in an agent; and relationships—the

[9][14] include the user model, the domain model (a model of the system), and an interaction model (a model of the user-system interaction).

way agents interact with one another. Her approach makes explicit the reasoning about the purpose of adaptations, treats human and computer agents the same in the environment, takes into account user motivation, emotions, and moods, and presents a unified model of collaborative, cooperative, and adverse behavior. The research presented in this dissertation addresses many of her concerns.

3.2 Elicitation of User Models

Elicitation of user models is a knowledge acquisition process. It is well-known that knowledge acquisition is the bottle-neck of intelligent system design. However, unlike knowledge acquired for an AI system such as expert systems, the "life" of a user model may be short-lived. Some user models are only useful for the duration of a problem solving "exercise" (e.g., student modeling and/or decision support user models [38]). The purpose of the user modeling component in an intelligent system has a great affect on the elicitation techniques used to construct the user model.

Knowledge in a user model may be acquired either implicitly via inferences made about a user, or explicitly via a collaborative process with the user, or a combination of both. Both acquisition approaches may be done before using the system [80], "on-line" while using the system [23, 68], or "off-line" after using the system. Determining when and how to elicit the user model knowledge is a domain and application dependent decision. For example, an application that will be used by a small group of users with well-defined input and output (e.g., accounting applications) may use a user model declaratively defined during the design of the application. Knowledge could be acquired using standard knowledge elicitation techniques [39]. On the other end of the spectrum, an application used by a diverse group of users (e.g., web based applications, word processors, decision support systems) with ill-defined and/or dynamic input and output must be capable of adapting to the needs of the user and the task at hand. Therefore, the user modeling component in these applications must be dynamic, adapting as the user interacts with the application. The next section discusses several different machine learning techniques with respect to their usefulness in various domains.

3.2.1 Machine Learning Techniques for Elicitation

[80] explain how detailed (and time consuming) user studies can be used to better understand the needs and behavior of users as they encounter problems using software applications. This understanding can then be translated into knowledge for use in a user model. Unfortunately, not all user model designers have the resources to be able to elicit user models by watching users perform their work. Machine learning techniques are the most common and widely used methods of eliciting user models implicitly from users. The main reason for the popularity of machine learning techniques for user model elicitation is the fact that the user model may be elicited incrementally, typically without user intervention by watching users performing their tasks. Therefore, machine learning techniques attempt to avoid the bottleneck of knowledge acquisition for user models.

Perhaps one of the most popular heuristics for elicitation is statistical correlation. That is, actions a user performs more frequently, given a "situation," are good candidates to try again given the same situation. Due to its popularity in the research, the next several paragraphs discuss several key pieces of research related to statistical correlation.

[105] uses memory-based reasoning [152] to learn *situation-action* pairs in an electronic mail application, meeting scheduler, and Web news reader domain. Memory-based reasoning is a frequency-based machine learning technique. Situations are composed of vectors of features relevant to the

environment the system is situated within. For the e-mail application, this might include the sender and recipient of a message, "`Subject:`" line keywords, the message length, the message urgency, etc. The relevant features to be included in a situation are typically determined during the system design phase. As the user interacts with the application, situations (defined by the features) and the actions taken by the user for a given situation are recorded. An *interface agent* uses the raw situation-action pair data to attempt to offer assistance to the user based on a closest match to previous situations. The frequency of the situation-action pairs determines their applicability. The statistical correlations are updated off-line. [105] uses two thresholds to determine when the statistical correlation is "strong" enough to warrant interface agent action on behalf of the user.

Bauer investigates the acquisition of user preferences for plan recognition, in a Web news reader domain [13]. He uses an off-line ID3 decision tree inductive learning algorithm [136] to learn classes of situations based on user actions. Bauer states his approach is better than the approach used by [105]. In particular, memory-based techniques maintain only the "raw data" in a situation-action database, while inductive techniques attempt to draw higher level abstractions from the situations. Additionally, situation-action pairs are restricted to one-step predictions of single user actions, preventing the direct application of situation-action pairs for plan recognition, where the evidence has accumulated over several observation steps.

Several other situation-action pair deficiencies exist. First, determining the relevant features to model the situations is critical. Within a restricted domain such as news readers, this is a tractable problem. However, for many domains, this restriction makes the situation-action pair technique inadequate. This problem is not unique to situation-action pairs[10]. Furthermore, to match an action to a situation, the situation must be fully specified. In memory-based reasoning, each feature in the situation vector is relevant by default. User model designers must be able to specify the situation unambiguously. Due to inherent uncertainty in many environments, due to faulty sensors, unobservable factors, etc., designers may not be able to specify a situation vector unambiguously. Second, the user must perform a number of actions prior to making any decisions. This problem can be overcome by using additional elicitation techniques such as user profiles or stereotypes providing *a priori* knowledge about a user. Third, most current situation-action pair-based user models ignore the *utility* of offering assistance to a user, assuming frequency is *the* key determining factor of user behavior. Systems must not only be capable of determining the frequency of an action given a situation, but the relevancy based on many factors, to include the goals the user is pursuing, cognitive factors such as spatial and temporal ability, skills proficiency, etc. "Situations" typically do not capture such factors. Fourth, situation-action pairs ignore the correlation between actions (i.e., behaviors) and the goals being pursued. Situations are matched with a single action to perform, given the situation. However, in many situations, a user may perform a series of actions to accomplish some higher level goal. Simple situation-action pairs cannot model situation-goal-actions. Lastly, while most user models using situation-action pairs allow for machine learning techniques to extend the behavioral model of users by adding new pairs and/or updating the statistical frequencies, they fail to capture the inherent uncertainty and dynamics of predicting the user's intent.

[10]Bauer states "the question of how to come up with an appropriate feature set for the description of situations can be crucial."

3.2.2 Elicitation Examples

Nothing substitutes for knowing the users. Several studies have been conducted in the domain of web based search. [108] discuss a model of what people do when they search for information on the web. Their model is based on the observations of seven experienced web users in answering specific questions via web based search. Their data shows that (a) each individual has a standard pattern of search behavior; and (b) when an individual deviates from the standard patterns, he or she recalls the search fitting a standard pattern. Participants recalled and relied on only a few of the sites they visited. Maglio and Barrett term these *waypoints*. Waypoints are easily recognized as lying along the path to a goal despite the fact that the same path is not followed to a goal in every case. Based on their findings, Maglio and Barrett have constructed Web agents that model users waypoints. The agents elicit user models by identifying repeated hyperlink-following patterns and waypoints used in finding information. The agents assist the user by suggesting similar patterns in similar contexts and help the user visualize the various waypoints.

[146] discuss the use of human intermediaries for information retrieval in large databases. They categorize utterances and elicitations from users and intermediaries into eight categories. Table 2 outlines the categories.

Table 2: Categories of Utterances and Elicitations (condensed from Saracevic et al. (1997))

Category	*Description*
Context	User's problem or task at hand
Terminology and restrictions	Elaboration on and modifications of concepts, terms, keywords, and descriptors
System explanations	Workings and technical aspects of system used
Reviews and relevance	Review of search statements with respect
Action	Description of an ongoing or impending activity
Backchanneling (prompts and echoes)	Communication prompts, fillers, acknowledgments, formulaic expressions, etc. indicating listeners involvement
Extraneous	Utterances extraneous to the search interaction

The data presented by [146] sheds some light on the research field of user modeling. Search tactics and procedures comprised one fourth of all utterances in the user-intermediary discourse. Intermediaries talk about twice as much as users in this category. Conversely, and very surprisingly, utterances dealing with context were the fewest. As [146] point out, this data challenges the usual assumption that user modeling largely involves modeling context. Backchanneling (e.g., "O.K.", "What?", grunts) represented the second largest category of utterances. This feedback is integral to human-human communication but seems largely ignored by the user modeling community. Elicitations composed one tenth of all utterances, with intermediaries making 60% of the elicitations. This data shows a reliance on the question-answer process to perform user modeling. The remaining 40% of elicitations made by users indicates their desire to understand the search process.

[45] determine the most plausible user model via an abduction recognition process. [45] regard the elicitation of a user model as a recognition task. Their approach uses an explicit collaboration with a user, knowledge of a user's run-time environment, and observation of the user's interaction with the system [46]. For the collaboration, users perform simple form-filling operation to elicit interest metrics for "intent-based" authoring. A user's run-time environment might include data files available on the host system the application is executing on, such as `.plan` files or pre-existing user profiles. Additionally, a user's goals and plans are inferred by observing his actions with the system's interface. The most plausible user model is dynamically constructed based on the explicit and implicit information provided by the user.

3.3 Specification and Design of User Models

Determining how to specify and design a user model to accurately model a user is difficult at best. The work done in artificial intelligence knowledge representation research is of particular use here. Many research interfaces use rule-based intelligence [160]. Rule-based representations, like those used in most intelligent user interfaces, fail in two key areas: representing uncertainty and dynamic user modeling. The use of "probability modules" [166] is an *ad hoc* approach to determining answer reliability, i.e., uncertainty. Furthermore, the addition and deletion of rules to dynamically model a user is *ad hoc*. Therefore, knowledge representations that can dynamically capture and model uncertainty can improve the user modeling in an intelligent user interface. In recent years, numerical uncertainty techniques from the AI community have been used in a number of systems to capture the uncertainty inherent in modeling users [82].

In general, two main schools of thought exist on the design of user models. The first uses "hand-coded" user models. That is, the system designer determines how best to model the users *a priori*. (See [82] and [104] for examples). Hand-coded user models are typically static. Once they are designed, they will not change structure. The second method uses "machine-coded" user models. That is, a user model is constructed by the system as it "learns" more about the user. These models are dynamic (i.e., the structure and/or "contents" of the user model changes over time). (See [82] for examples.) Both methods have advantages and disadvantages and the particular domain will determine which method best meets the needs of the user model's construction. It should be obvious that the elicitation method(s) used may be integrally tied to the way we specify and design the user model.

3.3.1 Relevancy and User Intent

As mentioned previously, to accurately predict user intent, we must have an accurate cognitive model (i.e., user model). Modeling every possible naturalistic property in the user's world fortunately does not lead to the most accurate model [49]. If this were not the case, we would have little hope in using numerical uncertainty management techniques such as Bayesian networks due to the computational inefficiency of large networks [41].

DeWitt notes that not all naturalistic (i.e., observable) properties play an interesting role in a user's causal model [49]. That is, only certain observable actions and information in a user's "world" will have relevance to that user. DeWitt uses the term *causally efficacious* to describe naturalistic properties that "play any interesting causal role in cognitive functions." He argues that not everything observable is of interest when we make decisions. DeWitt's philosophical argument has direct analogy in user models. A user model must not include every possible piece of information about "the world." To use DeWitt's example, while the manufacturer of a stereo, the amount of dust on top of the stereo, and the weight of the unit are all observable properties, none of this evidence would likely have any bearing on the reasoning as to why the stereo does not work. To include such information in a user model needlessly complicates the model, not only semantically, but computationally. To take this analogy one step further, the less causally efficacious a property is to another, the more likely it can be ignored. Therefore, to effectively and efficiently capture user intent, the user model should not attempt to model every possible action the user may exhibit, but only those that are relevant.

[75] use the term *relevancy set* to describe those properties included in their user model, represented by an *interface learning network.* The interface learning network is a Bayesian network supplemented by information support nodes storing the statistical frequency of user actions. For hand-coded models, the relevancy set will not change. For machine-coded models, the relevancy set may change with use. A *relevancy neighborhood* are those properties that are immediately causally efficacious to the decision under consideration. As an example from [75], a user model may contain the possible communication modes and tools a user may use in a system, given the user's class and individual preferences learned by the user's interface learning network, as well as several knowledge bases used previously. These nodes in the network are considered the relevancy set. When the interface learning agent wants to predict if a particular knowledge base will be needed by the user, the other knowledge bases are in the relevancy neighborhood. Therefore, the relevancy presents a computationally efficient and semantically meaningful view of the user's relevancy set at any given time. The concept of a relevancy set is a good tradeoff between computational complexity and representational exactness, while maintaining full semantics.

3.3.2 Specification and Design Examples

Various types of models of the user can be elicited. Stereotypes [139] represent "typical" users, expressing various traits, characteristics, and attributes that are common among a group of users. Users classify themselves as belonging to a particular stereotype. User profiles are used to represent background, interests, and general, typically static, knowledge about a specific user. User profiles may be elicited from users by, for example, standardized tests, surveys, and/or assessments. These elicitation techniques are used to construct the user profile.

User modeling researchers, as well as other research disciplines, have used stereotypes and user profiles for various purposes [89]. User modeling examples include using stereotypes to filter World Wide Web documents (HTML/text), basing a user's inferred interests on user adapted stereotypes acquired via experts [3], to derive initial proficiency estimates on a user's level of advancement for use in computer-assisted language learning [120], and adapting Web-based hyper-media based on stereotypical users' abilities (e.g., disabled and the elderly) [54].

As an extended example, [46], as well as [45], use a simple form-filling operation to elicit interest metrics for "intent-based" authoring. Simple Horn-clause dialect [134] is used to represent the contents of the knowledge-bases that compose their user model. Use of Horn-clauses allows inferencing to be performed using a best-first probabilistic Horn-clause assumption based system. The user model is represented by a set of recognition assumptions that satisfy the observations of the user's actions. The abductive reasoning engine is used not only to determine the most plausible user model, but the best (i.e., least costly) presentation of the information in the system.

As mentioned previously, modeling users involves a great deal of uncertainty. Using knowledge representations capable of dealing with this uncertainty in a sound, mathematical way is desirable. [82] provides an excellent overview of the use of numerical uncertainty techniques in systems utilizing user modeling systems. In particular, he focuses on Bayesian networks [130], Dempster-Shafer Theory, and fuzzy logic knowledge representations. In addition to categorizing each system by its knowledge representation, he also categorizes each system by its input (termed observable states and events), long- and short-term cognitive states, and domain. Furthermore, he critically discusses the advantages and disadvantages of each knowledge representation from the viewpoints of knowledge engineering requirements, programming effort, empirical model adjustment, computational complexity, "human-likeness," justifiability, and explainability.

One of the most widely used applications employing an uncertainty knowledge representation user model is the Microsoft Office '97 applications. [80] present their work on the Lumiére project. The Lumiére system utilizes a Bayesian network user model to capture the needs and goals of users within the domain of the Microsoft Excel spreadsheet application. The Bayesian network is elicited from application designer experts who had the advantage of performing "Wizard of Oz" studies[11] on real world users. To aid designers, a Lumiére Events Language was developed that allowed atomic application events to be directly modeled, as well as streams of atomic events to be formed into Boolean and set-theoretic combinations of low-level events. Furthermore, the language allows designers to compose new modeled events from previously defined modeled events and atomic events. The language allowed the researchers to build and modify transformation functions that would be compiled as run-time filters for modeled events.

[80] discuss the framing, constructing, and assessing of user models. The "Wizard of Oz" studies revealed classes of evidential distinctions, providing observational clues valuable to making inferences about a user's problems and the user's need for assistance. The identified classes are as follows:

- **Search**—Repetitive, scanning patterns associated with attempts to search for or access an item or functionality.
- **Focus of attention**—Selection and/or dwelling on artifacts in the interface.
- **Introspection**—A sudden pause after a period of activity, or significant slowing of activity.
- **Undesired effects**—Attempts to return to a prior state. This includes undoing actions, or closing a dialog box shortly after it is opened without invocating an offered operation.
- **Inefficient command sequences**—Performing operations that could be done more simply or efficiently.

[11] "Wizard of Oz" studies involve users interacting with applications (e.g., Microsoft's Excel) while a human expert provides application assistance to the users. The users are unaware the assistance is being provided by human experts. The effect is a person "behind the curtain."

- **Domain-specific syntactic and semantic content**—Consideration of special distinctions in content or structure of documents and how a user interacts with these features.

[19] notes when a user model designer attempts to determine what is important in the domain to model, he/she uses one or more heuristics, or general rules of thumb, to guide his/her knowledge acquisition task. These heuristics attempt to capture salient characteristics that define user intent by discriminating certain observations as being relevant to determining the user's intent. These heuristics are not mutually exclusive and designers of user models typically use more than one heuristic when designing the user model. As such, we must identify the common characteristics of user model elicitation and adaptation techniques. User models may be adapted as a result of applying one or more heuristics. Relevant heuristics identified by [19] are as follows:

- **Causality**—*Why* a user performs actions. The user intent ascription philosophy presented in the introduction states that a user perform actions in response to environmental stimuli and to achieve some goal. As a concrete example, [82] describes how a causal planning model can be used to construct Bayesian networks.

- **Context**—*What* is the current context. For certain types of interface agents—most notably agents for information filtering and/or data mining—the current context is important. Taking an example from [62], if a user is referring to a bank, it is useful to know whether he/she are referring to a financial institution or a river bank. The context of previous interactions may help disambiguate the current use of a word.

- **Frequency**—*How often* a user performs an action. Some interface agents use a fading function so actions become less relevant as time progresses.

- **Human-Factors**—*Who* is the user. Knowing user information *a priori* can be useful for adapting the interface to the user's needs. Human-factors such as psychological factors (e.g., spatial ability, cognitive ability, temporal ability), as well as physiological factors (e.g., skill level, age) may be directly applicable to the user's needs.

- **Modality**—*What* modes a user prefers, or uses explicitly or implicitly. This heuristic captures a large portion of the meaningful characteristics in direct manipulation interfaces. For example, what skill level (expert, intermediate, novice) does the user prefer? What type of ways do they like to view their information (e.g., full page, page layout, outline)? What about presentation methods such as textual, graphical, or audible? Do they prefer natural language or not? The specific "tools"[12] a user prefers to use can be considered a mode of the application. The orientation, size, position, etc. of the various windows, icons, and menus can also be considered a mode.

- **Resource Usage**—*What* resources a user needs. A resource can be as simple as a file or printer, or it can be another application, such as a World Wide Web search engine. As an example, [169] predicts Unix resource prediction using graph-based induction.

[12]Tools are defined to mean any function that helps the user get his/her job done. A tool can be a spell checker in a word processor, a macro, or a meta-method of activating another tool, such as using hot-keys, pull-down menus, and/or icons.

- **Temporality**— *When* a user performs an action. Do they perform a sequence of actions upon starting the application or prior to exiting? Does a user always react/respond a particular way to a certain action? The actions may be executed by the interface agent on behalf of the user.

[88] present an approach to determine, within the domain of intelligent multimedia presentation systems (IMMPS), what, when, why, and how, to adapt the system's presentation. Central to their approach is an explicit decomposition of the adaptation process into adaptivity constituents (the "what"), determinants (the "when"), goals (the "why"), and rules (the "how"). Their approach does not address agent-based environments, although the authors are outlining a project concerning the implementation of their approach within an agent-based environment[13].

The authors' work has several weaknesses. [88] have no way of determining whether their method of adaptation, the "how," is feasible within their approach, nor its impact. The authors' approach is to rely on the application to determine whether the adaptation method is feasible[14]. This places an unnecessary burden on the application designer to account for this. Furthermore, it makes integration of agents into legacy software nearly impossible. From a computational standpoint, certain adaptations could be abandoned given the evidence that the approach would not be feasible. All goals within a system deal with adaptation of the presentation. These are not necessarily explicit user goals. Therefore, the assistance offered may not help the user achieve a goal they are pursuing directly, but may indirectly help them by presenting the information in the "best" (as determined by the designer) way.

[169] uses a directed graph to represent the user's use of files and other system resources in the UNIX operating system[15]. The author states that user models acquired by automating a user's sequence of commands do not always typify the user's behavior. [169]'s work investigates the importance of data dependency between commands the user invokes. Experiments show that taking into account data dependency yields a better user model and improves command selection accuracy. [169]'s framework involves using the directed graph to model relationships between the commands a user invokes and the files needed by those commands. The graph representation is particularly well-suited to multi-tasking environments such as UNIX. As the user interacts with the operating system, the commands and input/output access is "recorded" in the directed graph. This graph is then analyzed via graph-based induction. Typical subgraphs are extracted from the input graph so that the subgraphs represent typical events in the system. These extracted subgraphs are the user models and are used to pre-fetch needed files based on prediction of future commands of the user.

[125] uses the concepts of finite-state grammars to generate command languages. The author states that command languages have some characteristics of certain finite-state grammars implying command languages can be formulated and represented by production rules generated by a finite state grammar. The author also states that communicated intents, driven by performing activities, have properties similar to command languages. The usefulness of this representation comes in the prediction and adaptation to users by representing their intents in a knowledge base and inferencing over this knowledge base to determine what the user is attempting to do. The knowledge representation works well in application domains where there is little ambiguity concerning what a particular activity might convey in the way of intent; i.e., domains where tasks are well defined.

[13]Personal communication with Constantine Stephanidis.
[14]Personal communication with Constantine Stephanidis.
[15]For a more detailed presentation, see [170].

[153] discusses how designers can specify and design intelligent user interfaces through the identification of human task and work flows. [153] presents the TADEUS (Task Analysis, Design, End-User Systems) approach, a task-oriented system development approach. His approach deals with designing intelligence interfaces by addressing what a user needs for task accomplishment. This is in contrast with using machine learning techniques to elicit "intelligence" although his approach does not preclude using machine learning techniques. Using the TADEUS approach, work flows are migrated into the user-interface design representations. Business goals and rules and their relationships to tasks and people involved can be integrated into the user interface design. Meaningful sequential activities for task accomplishment are determined. Profiles of various users of the system, and their functional roles to the task and work flows can be generated. Flow and control of data can be made transparent and interface designers can provide notation that allows static and dynamic specification and adaptation of the work flow models. The TADEUS approach allows consistent and context-sensitive user interface development to be supported in the software development process and eases the use of artifacts of previous development.

3.4 User Model Verification and Validation

Once the user model has been elicited, specified, and finally designed, determining if the user model is an accurate representation of the user's actual interaction with the system and meets all designer requirements is paramount to the functionality of the user model in a system. The accuracy of the user model must be measured if we are to determine how closely the user model matches the real (exhibited) behavior of the user. Since the user's behavior may change over time, deviating from the the originally elicited, specified, and designed user model (assuming we accurately captured the user's model initially), verification and validation (V & V) techniques must be used *over time* to make sure the user model currently reflects the user's behavior.

The difficulties in the development of user models, particularly with regards to the knowledge representation and knowledge elicitation techniques used, often leads to errors in several forms. Imperfect information is ubiquitous—almost all the information that we have about the real world is not certain, complete, or precise [129]. Three concepts that are essential for V & V of user models are incorrectness, incompleteness, and inconsistency. Incorrectness—having the wrong information—occurs when the system using the user model does not "behave" in the manner the user thinks it should. Finding the location of the wrong information can be difficult, and correcting it even harder. This aspect of validation can be addressed through a variety of approaches discussed earlier in this section (i.e. machine learning techniques).

Inconsistency—having conflicting information—is related to incorrectness. Where incorrectness is having the wrong information, inconsistency is the result of either having the wrong information at the wrong time or the right information at the wrong time. Inconsistency in a user model is primarily related to the inability to model the right information at the right time. For example, a user model for a World Wide Web browser may model a user's preference for text-based search results versus graphical-based, but fail to model the user's preference to view the results graphically when the number of results exceeds a certain threshold. In this example, the user model has failed to model the context of the number of results and the user's threshold. These types of errors are often discovered and corrected within the user model elicitation process.

Incompleteness—not having the right information—exists when a set of input values (e.g., observations) is passed to the system and the system fails to arrive at a proper conclusion. This type of omission can be very difficult to detect and locate as well. User model incompleteness can be both intentional (e.g., information disregarded to make the user model inferencing tractable) or unintentional (e.g., an oversight). Incompleteness can come from several different sources. Human error is often the major source for this type of error. Experts often have difficulties in conveying complete heuristic knowledge to the knowledge engineer. This lack of information often leads to incompleteness in the user model. Often information is missing during development of the user model and is left out for future modifications. Other types of knowledge are yet to be discovered. For these and other reasons, the ability to handle incompleteness is critical in the validation of these systems.

V & V involves issues such as exploring the completeness of the user model, accuracy of information, and the more sophisticated problem of dynamic interactions between knowledge chunks in the decision-making process. While the main objectives of V & V are closely knitted together, it is important to understand the distinct differences between them. Verification is best defined as making sure the system[16] is built correctly. Critical to this step is ensuring all information deemed necessary is included and that this information is interpreted and applied correctly by the system being inspected. If specifications exist for a particular system, verification will check for compliance with these specifications. [26] explicitly take into account an agent's requirements and metrics for measuring the agent's effectiveness of meeting those requirements. Using these requirements and metrics, they develop a requirements utility function that determines *when* a user model should be corrected and *how*. Verification is often referred to as clear-box testing.

Validation, on the other hand, is used to ensure the output of the system is correct. It is also used to check the system developed is what the users requested. It must assume the user model was built satisfactorily. Unlike elicitation and verification of user models, relatively few or no methodologies exists for validation except to simply hand the expert the entire user model and the field-test results. Very little help is provided by the system to detect and correct the errors leaving the human experts on their own. Typically, validation consists of running a sequence of test cases through the system and comparing system results against known results or expert opinions [126]. This is a time-consuming process and never guarantees finding all errors, especially in larger systems. [126] state "validation can be considered the cornerstone of evaluation, since highly efficient implementations of invalid systems are useless." Validation is often referred to as black-box testing. Concern is placed not upon what is inside the system, but what the results are coming out of the system. Despite the importance of validation, the majority of V & V literature for knowledge-based systems is solely concerned with verification, specifically automatic rule-based error checking. This aspect of knowledge-based V & V has now become reasonably mature and many such automated tools exist [122, 113, 135]. This automation is often built into the system so that verification is continually addressed throughout user model adaptation to ensure a quality "final" product. Use of these tools and techniques used in user model adaptation fail to measure how closely the user model reflects the user's behavior. Typically, usability studies are performed for this purpose. Users are asked a number of subjective questions concerning their likes and dislikes of the system, and from these questions, user model and/or system designers ascertain the validity of the user model (for example, see [6]). Most usability studies are done "off-line" and have no immediate bearing on the user model.

Testing, including validation, is best done throughout the entire development of the user model.

[16]When we say "system" we broadly mean the user model and the system that contains the user model.

Incremental testing can aid in finding inaccuracies or incompleteness early in the development of the system rather than later when corrections can be much more difficult to detect, locate, and correct. In determining the overall validity of a system, it is often beneficial to determine how well human experts do in the problem area and to create reasonable expectations of the systems performance. Typically, user model performance can change drastically from initial release to later stages of use. Some systems can be field tested and validated in its early use without harm. In critical applications where lives may be at risk, field testing is not always possible. User model designers should maintain involvement throughout development of the system whenever possible. This can often assist in identifying errors early on in the development cycle that may not have been detected until later stages of validation.

For dynamic, "machine-coded" user models, verification and validation must be re-accomplished after each change. Verifying and validating after modifications or enhancements have been implemented is just as important as earlier testing. Testing needs to ensure that the original system was not degraded as well as that the modifications made were correctly implemented. Comparison of previous test case results and their performance after the modifications is an effective way of testing the updated system remains validated in areas both inside and outside of the modified areas.

This section has previously discussed a number of machine learning techniques for eliciting knowledge for use in dynamic, "machine-coded" user models. In addition to elicitation of new knowledge, machine learning techniques are useful for "on-line" verification of user model knowledge. That is, machine learning techniques provide a means to *implicitly* verify the user model and correct errors found. We say implicitly verify because most machine learning techniques for updating the user model are not grounded in a sound methodology for verification. We must determine *what is important* to model in the domain, with associated discriminators and/or metrics to determine and define *why*, *when*, and *how* to dynamically change the user model. To improve the user model-based system's ability to accurately model the user we must explicitly take into account the user model requirements. Objective metrics are useful for *dynamically* modifying the user model to better model the user [26]. [77] states interface agent designers need a better understanding of how intelligence can *improve* the interaction with the user. This understanding includes *how* machine learning techniques affect the user model and therefore interaction between the interface agent and user. Failure to understand this relationship may result in a fruitless search for ways to improve the user model.

3.5 Utilization of User Models

In practice, user models used in "real" systems make several assumptions, as described by R. V. London[17]:

- **Closed-world assumption**—All relevant plans and actions are known either explicitly or by closure.
- **User's correctness**—The user has a coherent, well-formed plan, with no fatal misconceptions.
- **Unified goal and plan**—Typically, the user has a single top-level goal, with subgoals in a proper hierarchy. If there are non-hierarchic goals, then they are either resolved into a unified

[17]As cited by [84].

plan or maintained as multiple plans that are nearly independent. Additionally, users have a somewhat persistent focus.

- **Co-operative user**—The user is purposely giving information to help the user modeler, or the system can verify and adjust its conclusions by cooperative negotiations with the user.
- **No real-time requirements**—The system is not required to respond within the time bounds of normal human communication.

[132] further discusses shortcomings of user models utilized in most systems:

- Assumptions about interaction preferences or behavior patterns are missing.
- Assumptions are acquired with specialized heuristics, drawing from isolated observations without regard to interaction context.
- User behavior, preferences, and mental attitudes are subject to change, but not adequately modeled.
- User models are constructed and exploited mostly within the limits of one application. However, it is beneficial to share information about users among several applications.

3.6 Plan Recognition

Plan recognition is the task of ascribing intentions about plans to an agent (human or software), based on observation of the agent's actions. There are three types of plan recognition: intended—the agent chooses actions that make the plan recognition process easier; keyhole—the agent is unaware of or indifferent to the plan recognition process; and obstructed—the agent is aware of and actively obstructs the plan recognition process [164].

Several researchers have used plan recognition as a user modeling technique. Plan recognition has been used to predict actions in a virtual predator-prey environment [56] and to predict users' actions in games [2]. Several authors have investigated the use of probabilistic approaches, including Bayesian networks for plan recognition and generation.

[92] investigate the use of a Bayesian network of the "world"—in the authors' case, a simple room with a mobile robot and a target to be found—and a utility measure on world states to generate plans (sequences of actions) with high expected utility. The number of plans investigated are restricted to those with high utility with respect to attaining a goal. In the authors' work, the goals are known with certainty and the plans to achieve the goals must be found. They show that by using decision-theoretic methods, they can drastically reduce the number of plans that must be investigated.

Waern uses keyhole plan recognition to determine what a user is doing within a route guidance domain and Internet news reader [164]. She uses pre-compiled plans called *recipes.* Assistance is offered for attaining a goal based on the calculated probability the user is pursuing the goal. Her approach does *not* address the utility of offering assistance. She considers only the probability that offering to perform an action will help the user obtain a goal. Furthermore, her approach is not dynamic; that is, there is no adaptation of the pre-compiled plans. Lastly, there is no way to *trace* the user's execution of a plan nor offer explanations of assistance.

ANDES is a system that performs long-term knowledge-assessment, plan recognition, and prediction of students' action during physics problem solving [38, 60]. ANDES uses Bayesian networks constructed dynamically from static knowledge bases of specific physics problems as well as physics concepts, laws, formulas, etc. The plan recognition component is used to tailor support for the students when they reach impasses in their problem solving process. Their approach uses explicitly stated goal and problem solving strategy nodes to model the user's problem solving, but does *not* use a utility-based approach. The authors note they are researching methods to expand the fixed knowledge base used to construct the Bayesian networks; in particular, they are adding knowledge to determine common "misconceptions" within the domain and changing the static parameters.

The Pilot's Associate Program was a research effort using plan recognition within a real-time domain to determine goals from actions [7, 71]. The Pilot's Associate was a software agent providing assistance—in the form of wanted and needed information at the correct time—to a combat pilot. The Pilot's Associate pushed the envelope in knowledge representation and reasoning techniques. They used AND/OR plan-and-goal trees with an associated "dictionary form" for explanations of plans a user was pursuing, and offered assistance to help the pilot. The graph's hierarchical structure produces a common planning language for use among heterogeneous modules. One of the main problems with their approach was due to the technology available at the time for representing and reasoning about uncertainty. As a result, their approach did not account for the uncertainty nor utility in obtaining goals by performing actions.

[82] presents an overview of a casual planning model that can be summarized in terms of the following phases:

1. A user observes aspects of the physical environment (e.g., stimuli).
2. The user compares the observations with his/her goals, determining the extent to which her goals are fulfilled.
3. The user selects a plan to address an unfulfilled goal.
4. The user refines a plan, taking into account the current situation.
5. The user executes a sequence of actions.
6. These actions have observable effects on the environment.

Plan recognition is made difficult because many times users do not follow pre-planned goals. They perform some actions that can be ascribed to more than one plan, and may be pursuing several plans at once. Because of this, ascribing intentions to a user's actions by observing the actions to those actions can be next to impossible. One way of avoiding this is to observe only the most recent actions [164, 56]. A *fading function* is used to "forget" past actions. Not only does this have the advantage of focusing attention on the most recent actions (making plan recognition in certain domains, e.g., web browsing) easier, but it has the side effect of reducing the complexity of reasoning over all the past actions to determine an agent's plan. Another way to handle this problem is to introduce uncertainty into the model. [82] describes how the causal planning model can be used to construct Bayesian networks. The networks are constructed based on observable events within each phase of the model.

3.7 Conclusion

The active user interface metaphor builds on the foundations laid by three separate research fields—human-computer interaction, artificial intelligence, and user modeling—concerning interface agent research. The orthogonal approach of each field presents a number of strengths. Furthermore, the relevant weaknesses of the three research fields have been given. These weaknesses have served to provide areas for additional research. While researchers in each field are cognizant of the strengths and weaknesses in their own field, this background has served to introduce researchers in any one field the strengths and weaknesses of the other two. This section has also served to summarize relevant background on plan recognition, as it relates to user modeling.

4 Active User Interfaces

Active user interfaces[18] are the next logical step towards addressing the information overload and under load problem. Whereas "interface agents" are designed around the assistant [105] and/or associate metaphor [7, 71, 116], active user interfaces serve as an *actuator* to the human-machine interface, allowing the user to interact with the computer in a naturalistic way. Many of the characteristics are exhibited by existing interfaces. However, the focus of the interface design shifts from "agent-centered" to a human-centered system.

We first provide and discuss our motivation for active user interfaces. Next we discuss active user interfaces and the new user interface metaphor they provide for user interface design. Finally, we provide two examples of first-generation active user interfaces within the domains of information management and decision support systems. We discuss how such interfaces are capable of providing an intuitive, meaningful, and useful "actuation" interface for a heterogeneous collection of information sources and tools.

4.1 Motivation

The primary goal of the active user interface metaphor is to reduce task overload and under load. This new metaphor arises out of the active debate between the agents and direct manipulation interfaces research communities. Active user interfaces share many ideas from the user modeling and interface agents research fields but also draw heavily from research in the human-computer interaction and artificial intelligence research fields. The resulting synergy of the techniques, methodologies, and philosophies of these fields can provide a powerful approach to designing interfaces.

To accomplish the primary goal, a comprehensive software engineering, knowledge engineering, and knowledge elicitation approach for software systems must be developed that is inclusive of the human element involved. We emphasize a paradigm of highly interactive collaboration between the human expert and the machine.

To briefly recap, a strength of HCI research in interface agents is its attentiveness to the user, focusing on how best to represent information to the user. A strength of AI research is its years of experience in knowledge representation, reasoning, and machine learning. The strength of user modeling is its ability to capture the user's knowledge (e.g., domain knowledge, skills, preference, goals/tasks) and interaction within a system to adapt those systems to the needs of users. From the HCI perspective, the approach must focus on the *interactions* of the user and system, determining what interactions must occur, when the interactions must occur, how the interactions are to be presented, and why to present them. The HCI perspective is one devoted to helping users concentrate on what is important in the domain. From the AI perspective, the approach must focus on efficient and effective *representations* for domain knowledge and user modeling, determining what, when, how, and why to adapt these representations to improve the efficiency and effectiveness of the system. This includes representing and performing adaptation methods for the knowledge and focusing on abstracting and analyzing the relevant data. From the user modeling perspective, the approach must focus on the dynamic *adaptations* of the system to empower a user to perform his/her tasks, given his/her user model.

In active user interfaces, the interface should operate symbiotically; that is, work tasks should be appropriately partitioned between the computer and the user. The computer's strength lies in

[18]Different from active interfaces described earlier in Section 2 [30, 32, 31].

its ability to perform data acquisition and management (to include display of this information) from many heterogeneous sources, low level quantitative and qualitative data analysis, and routine inference to enable decision support. A user's strength lies in the ability to provide guidance and insight concerning the information that is necessary to draw complex, higher level inferences from the data. A symbiotic approach is necessary because the objective is to let the user and the computer share the task load; therefore, a human-centered approach to task partitioning is used.

Active user interfaces are not only concerned with "shielding" users from the complexities of information sources but they also focus on the use of information visualization techniques to enable the user to understand the derived information, synthesis operations, and available processing options. However, to maximize user effectiveness in an information dense environment, the approach must operate in anticipation of user information needs. To do so, we must first ascertain user information requirements based on the current situation and a history of required information. The approach then initiates data retrieval operations and provides focus of attention on and analysis of the resulting relevant information.

The active user interface approach requires the development of an adaptive, intelligent, learning human-computer interface. Construction of the interface requires a mix of traditional human-computer interaction techniques, data visualization, and intelligent agents within a software engineering framework. To be sure, the intelligent agent paradigm is a key aspect of the approach. Intelligent agents perform information fusion, analysis and abstraction, as well as deriving information requirements and controlling information display.

An intelligent interface agent, possessing knowledge of the environmental stimuli, user goals, actions, and various human factors can serve as an actuator, providing assistance on behalf of the user. This knowledge can be captured in a user model. Specifically, a knowledge representation capable of capturing the causal relationship between the stimuli, goals, and actions must be used. Furthermore, since ascribing user intent is inherently uncertain, the knowledge representation should be capable of representing the uncertainty in a sound, non-*ad hoc* manner. As mentioned previously, [82] provides an overview of the use of numerical uncertainty techniques (i.e., Bayesian networks, Dempster-Shafer Theory, and fuzzy logic) in systems utilizing user modeling systems.

4.2 The Active User Interface Metaphor

Active user interfaces provide a new way of looking at user interfaces. These interfaces are capable of multi-levels of collaboration and autonomy. [124] states that user must feel in control and the system must provide accurate expectations (as well as minimizing false hopes). The user of an active user interface is fully aware of any actions, whether explicit (authorized consent) or implicit (implied consent), taken by the interface and has a complete, intuitive understanding of such actions. This "awareness" is provided by the active user interface by a variety of means, including explanatory capabilities, advanced visualization, etc.

Current work in interface agents follow the "assistant" and/or "associate" metaphor. The "assistant" metaphor is one in which the agent acts as a tool for the user to manipulate to perform tasks on his/her behalf. Examples include the Internet Softbot [52], Calendar APprentice [117], and Lumiére [80]. The "associate" metaphor is provides a more autonomous view of the human-computer interaction, where the computer is delegated certain tasks implicitly and within that scope of control, can act on the user's behalf. Examples include the Pilot's Associate [7, 71], CiteSeer [17], and Information Lens [109].

One of the problems with both metaphors is the issue of control over the computer's actions. In the active user interface metaphor, this issue is minimized by insuring the user is cognizant of the agent's actions at all times. The active user interface metaphor is one of an *actuator*. An actuator is a mechanism for controlling something indirectly instead of directly, e.g., by hand. By analogy, an active user interface is a human-computer interface where the user indirectly manipulates and controls the system artifacts. At first, this does not appear to be different from other metaphors, most notably the venerable direct manipulation interface. The difference is that with the direct manipulation interface, the user has a more tacit understanding that, for example, pushing the "summation" button in a spreadsheet will add the column of highlighted numbers. [40] calls this sort of interface the technology interface paradigm. On the other hand, active user interfaces are useful for systems where the interface and the abstraction of the underlying datum is difficult, if not impossible, to represent with "standard" direct manipulation interfaces. These non-WIMP (windows, icons, menus, and pointers) user interfaces are not based on the desktop metaphor. There are at least four basic non-WIMP styles: virtual reality, embedded interfaces, notebook, and hyper-media [66]. An example is Etzioni's web bots [52].

The following characteristics[19], summarized in Table 3, are central to active user interfaces:

Table 3: Active User Interface Characteristics

Characteristic	*Description*
Adaptive	The AUI modifies its internal representation of the world.
Adaptable	The user can modify the AUI's internal representation of the world.
Autonomous	The AUI "takes the initiative" without direct intervention from the user.
Actuation	The AUI affects its environment.
Attentive	The AUI observes the user and environment.
Assistive	The AUI provides assistance to user, helping him/her achieve goals.
Analytic	The AUI reasons about the user's actions and reaction as well as its own actions.
Auditable	the AUI "justifies" its actions by providing explanations of why an action was taken.
Addressable	The AUI converses with the user.
Amplifiable	The AUI is extensible to many domains.

- **Adaptive**—The active user interface possesses the ability to modify an internal representa-

[19]Some of these characteristics are included as requirements of an interface agent as defined by [26].

tion of the environment through sensing of the environment in order to change future sensing, acting, and reacting for the purpose of determining user intent and improving the *accuracy* of the assistance offered. The active user interface's ability to sense its environment implies it is *perceptive.* That is, it can distinguish relevant features in the environment in relationship to the method necessary to act and react within the environment. *Reactive* "behavior" implies a timely response (i.e., stimulus-response) to sensed events, whereas to *act* connotes a deliberative (reasoned) response to events. The relationship between the agent's sensing, acting, and reacting with regards to the internal representation of the environment further defines properties for the interface. [64] defines these deliberative (agent) properties to include *predictive*—the ability to model the environment so as to predict how its actions will affect the environment; *interpretive*—the ability to correctly assess its sensors; and *rational*—the ability to perform actions to obtain its goals.

- **Adaptable**—The user is able to explicitly adapt the interface to his/her own preferences. Whereas adaptive interfaces may alter the behavior of the interface without explicit consent of the user, adaptable interfaces are ones in which the user is in complete control of the adaptations. Examples include choosing a font size in a word processor and locations of "detachable" menus.

- **Autonomous**—One of the main strengths of agents is their ability to sense, act, and react over time within an environment without direct intervention. Direct intervention means explicit "activation" by the user. That is, they are capable of autonomous behavior. Active user interfaces extend this notion by considering the many possible levels of autonomy that exist. Table 4 (from [55]) present scales of degrees of automation. [9], [111], and [112] discuss how to represent dynamic autonomy within a system and support multiple levels of autonomy.

  [131] notes autonomy is not well defined within the agent community. [57] argue autonomy implies a reactive (sensing and acting within a time constraint), temporally continuous, and goal-oriented (pro-active) agent. Direct intervention means explicit "activation" by the user or other agents. Autonomous behavior does not preclude the active interface from responding to this sort of collaborative interaction with the user and other components of the system. Autonomy implies some sort of saved internal state, whereas adaptivity explicitly requires it.

- **Actuation**—The interface needs to be able to affect its environment via an actuation mechanism (e.g., "marionette strings" [102, 114]) to the systems that compose the environment. Concerning the interface's ability to act and react, the adaptivity characteristic assumes the interface is able to affect the environment through its acting and reacting.

- **Attentive**—The interface observes the actions of the user by "looking over the shoulder" [105] of the users. This indirect intervention approach allows the interface to observes what the user is doing without explicitly inquiring from the user what they are doing. This characteristic does not prevent the interface from questioning the user and other agents about their beliefs, desires, and intent especially in light of possibly conflicting observed actions [24].

- **Assistive**—The interface provides assistance to user, helping him/her achieve goals. The level of the assistance is related to the level of autonomy given to the active user interface to help the user (see Table 4).

Table 4: Scales of Degrees of Automation (from Flach and Kuperman 1998)

Category	*Description*
1	The computer offers no assistance; the human must do it all.
2	The computer offers a complete set of action alternatives, *and*
3	Narrows the selection down to a few, *or*
4	Suggests one, *and*
5	Executes the suggestion if the human approves, *or*
6	Allows the human restricted time to veto before automatic execution, *or*
7	Executes automatically, then necessarily informs the human, *or*
8	Informs him or her after execution only if he or she asks, *or*
9	Informs him or her after execution if the computer decides to.
10	The computer decides everything and acts autonomously, ignoring the human

- **Analytic**—Active user interfaces are introspective. They reason about why a user performs actions in pursuit of goals within the environment. Furthermore, the active user interface reasons about its own actions and how it can improve those actions to support the user better [26].

- **Auditable**—Active user interfaces can "justify" their actions by providing explanations of why an action was taken. Users can look at all of the information the interface possesses to make its decisions. [140] present experimental results conducted within an expert system domain that a good mental model of a system's capabilities leads to increased user/system interaction and performance and graphic inference explanations leads to higher performance than textual explanations. [168] uses a combination of user modeling, adaptive explanation generation, and hypertext techniques to adapt explanations that are not clear or complete. [115] show how a system capable of justifying its decisions improves user trust, offers embedded training potential, and increase system use accuracy.

- **Addressable**—In addition to the interface's ability to justify its actions, users may converse with the active user interface. That is, the interface agent has the ability to address (communicate with) other components of the system, including the user, and the be addressed in a "language" that is understandable to the other conversant. For human users, this can mean natural language interfaces [110, 156, 36], but may also include other interface methods such as audio, video, and/or animation. The multiple levels of collaboration with all components of the system best differentiates active user interfaces from other types of interface metaphors. Concerning collaboration with the user, this collaboration may be as simple as making a suggestion to the user and asking if the suggestion was correct or not, or as complicated as observing the user's actions within the environment, attempting to determine the needs and intent of the user, and providing assistance at "appropriate" times. Multi-agent systems are particularly astute at collaborating amongst other agents. The active user interface possesses this level of multi-agent collaboration as well. Collaboration allows agents to increase their internal representation accuracy, resolve conflicts and inconsistencies within the representation, and improve their decision support capabilities [157]. Collaboration may also involve other non-human components of the system. Collaboration implies an agreed upon communication language and a commitment to use that language. Central to collaboration are the *behaviors* an agent can perform and the *protocol* in which they communicate those behaviors [48].

- **Amplifiable**—The active user interface is extensible to other domains. It is desired for the interface to possess the ability to easily adapt to information and requirement changes. The former is best dealt with under the adaptivity requirement (e.g., the ability to adapt to different users), while the latter is better dealt with during interface's specification. The active user interface has the ability to add and remove new sensor information or system components dynamically or be used in a new environments with different requirements. With this ability, comes a certain robustness, i.e. an ability to degrade its assistance gracefully. Robustness is required for active user interface since they, more than other interface metaphors, must be capable of gracefully degraded performance because they must interact with the most complicated of agents—users.

Not every active user interface will possess each characteristic, nor will they possess them in the same amount. During the comprehensive software engineering, knowledge engineering, and knowledge elicitation for an active user interface, those characteristics deemed most important for interaction with the user and reducing the user's task overload and under load will be brought to the forefront. Examples of two system that use the active user interface metaphor are given in the next two sections. Each system—a natural language query information management system and a probabilistic expert system shell—possesses a number of the active user interface characteristics.

4.3 The Clavin Prototype–An Active User Interface Example

A natural language query information management system was used as one of the test beds for the active user interface concepts discussed in this article. *Intelligent information systems* provide "just-in-time" knowledge delivery [37] for users. The key research issues in this domain include task tracking, automatic query formulation, and personalization [72]. This domain was chosen for two primary reasons. First, the system was envisioned to be used by a wide diversity of users. For this particular project funded under the auspices of the Office of the Secretary of the Air Force, these users can be stereotyped into one of several groups (e.g., battle field commanders, intelligence officers, program managers). Yet, individual users have preferences in how they interact with the system, as well as differing levels of abstraction of the information they need to accomplish their tasks. Second, the information provided to the users had to be relevant with regards to current user tasks. This implies not only providing the right information, but at the right time. Therefore, this domain provides a way to test the active user interface's various characteristics; specifically the ability to offer timely, beneficial assistance to users (the assistive characteristic) and to adapt that assistance to the different needs of users (the adaptive characteristic). The natural language query information management system tests the active user interface's ability to dynamically construct and adapt a user model over time as the environment (e.g., the information source being managed and/or the users' needs) changes over time.

The Clavin system is an intelligent natural language query information management system [143]. Clavin must maintain a dynamic user model of the relevant concepts in the user inquiries as they relate to the information sources. The primary goal of Clavin is to autonomously react to changes in user intent as well as the information sources, by dynamically constructing the appropriate queries relative to the changes identified. Clavin does this proactively and autonomous, finding relevant information the user needs (the autonomous characteristic). Clavin responds to users' natural language inquiries by forming intelligent queries to a database of potentially dynamic, heterogeneous information sources. Clavin maintains a dynamic user model of the relevant concepts in the user inquiries as they relate to the information sources. The user model allows Clavin to determine the relevancy of the various concepts in the domain, in order to properly address user inquiries. Clavin should be capable of proactively retrieving relevant information for the user based on the current inquiries and the user's previous history of inquiries.

The Clavin system architecture consists of four main components: The human language interface (HLI) component is composed of a commercial off-the-shelf voice recognition system and a natural language (English) parser. The voice recognition system is responsible for transforming a spoken utterance by the user into a natural language sentence. The natural language sentence is then parsed into an S-expression predicate logic representation (well-formed formulas) of the user queries by the natural language parser (NLP) component. The well-formed formula (wff) query is then passed to the interface agent. (An example of an uttered query, its wff representation, and the resulting representation in the user model is shown in Figure 1.) Once the active user interface component processes the query (see the discussion below), the query is passed to the retrieval engine. The engine parses the query wff and transforms it into a database query. Heterogeneous, possibly dynamic information sources are then queried by the retrieval engine. These information sources are represented to the user as one homogeneous information source. Results are then passed back to the interface agent for data visualization of the resulting query. If no results are returned, the active user interface can request the HLI to produce another possible parse (i.e., re-tag the parts of speech and generate alternative wffs) for the uttered query.

Ultimately, the active user interface is responsible for two key functions within the Clavin system: information filtering and proactive querying. The first occurs as a result of adding context to the spoken queries. The second occurs as a result of combining various relevant pieces of previous queries. For example, if the user asks "What causes Lyme disease?" the system returns information about deer tick bites; then the user's next query asks "What treats Lyme disease?", to which the system replies with information concerning a new Lyme disease vaccine. At this point, the user model contains information about the concepts "Lyme disease," "tick bite," "Lyme disease vaccine," "causes," and "treats." If the user then makes a inquiry about "what causes cancer?" the interface agent component not only retrieves information about the causation of cancer, but also proactively queries the database about possible treatments for cancer.

The following key issues arise within the context of the Clavin System for active user interfaces:

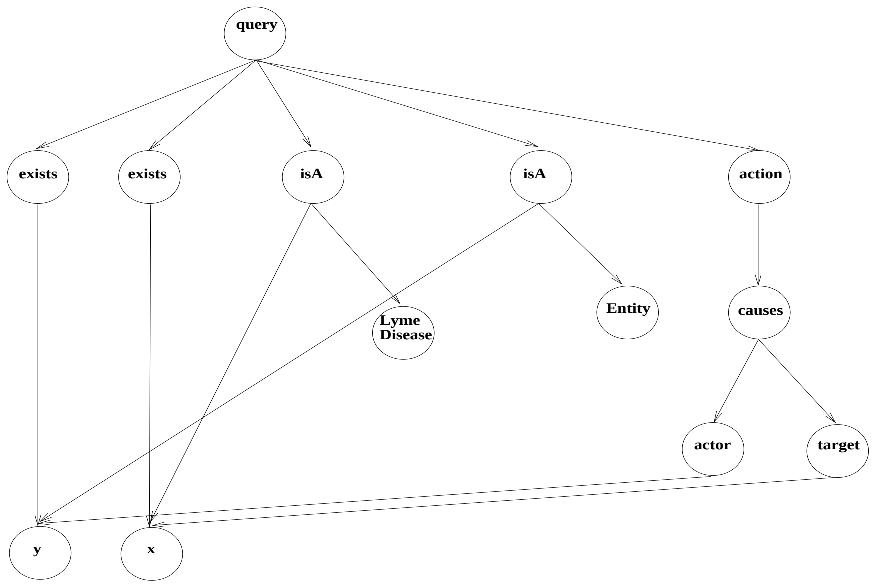

Figure 1: A User Model Representation of the Spoken Query "What causes Lyme Disease?". The utterance is transformed to the wff (exists x)(exists y)(isA x Lyme Disease)(isA y Entity)((action causes)(actor y) (target x)).

1. **"On-the-Fly" User Model Construction:** The active user interface approach offers a framework with which to determine which sub-queries are relevant and can be combined to make proactive queries on behalf of the user. By using the expected utilities of the various sub-query concepts (e.g., isA(x, dog), color(y, red), action(hit(john, ball))), the interface agent can combine those sub-queries with high expected utility. This approach allows the agent to combine concepts in meaningful, non-ad hoc fashion. This procedural-based user model construction method allows the active user interface to begin with no domain knowledge and incrementally construct the user model as the user interacts with Clavin. This unanticipated side benefit allows designers to mitigate the classic knowledge acquisition bottleneck problem.

2. **Agent Autonomy:** Since Clavin interacts with many heterogeneous information sources, some of them dynamic (e.g., news streams), the interface agent component of Clavin should be free to autonomously fetch new information, reason over it, and present updated results to the user. The issue becomes the following: how much autonomy should the agent have?

3. **Feedback:** While feedback from the user must be an integral part of the system for user intent modeling, the current approach of positing alternative queries and results to the user seems cumbersome. Is there a theory or model to better capture the users desires without a potentially lengthy trial and error approach?

4. **Dynamic Information Sources** Clavin must deal with dynamic, possibly massive information sources. Therefore a declarative approach to user model construction is not feasible. The sub-query relevance approach taken in Clavin offers a useful alternative to the declarative approach as well as data-centric, statistical learning approaches.

4.4 The PESKI Decision Support System–Another Active User Interface Example

Most everyday decisions involve some level of uncertainty. Expert systems, also known as decision support systems and knowledge-based systems, attempt to capture an expert's knowledge for use by non-experts. Among the advantages to using expert systems are wide distribution, accessibility, and preservation of scarce expertise [63]. One of the greatest disadvantages to expert systems is their construction. To aid experts in the arduous task of designing expert systems, a number of expert systems shells exist today. Most of these shells provide a variety of tools to the expert system designers to capture an expert's knowledge, verify and validate that knowledge, and query this knowledge, i.e., perform inference. However, these tools are designed primarily around the target knowledge representation, that is, the goals of these tools are to satisfy the constraints of the representation and guarantee that the user does so when modifying the knowledge. Furthermore, there is little or no agreed upon methodology for tools interacting with the human user let alone with each other, other than in a haphazard fashion. Thus, none of these systems provide an integrated (human and machine) approach for acquiring knowledge, testing that knowledge via verification and validation, and inference.

PESKI [21, 25] is a suite of intelligent knowledge engineering tools (agents) that have been developed and integrated using the active user interfaces metaphor. A primary goal of PESKI is to develop a comprehensive software engineering, knowledge engineering, and knowledge elicitation methodology for decision support systems also inclusive of the human element involved. Central to the approach are active user interfaces. Simply put, the ultimate goal for PESKI is to guarantee that any and all actions taken by the expert and machine in building a decision support system is done as efficiently as possible, always consistent, and always correct. Given that knowledge engineering is rife with many incremental choices and alternatives at each stage, making the right choices by the human/machine is paramount.

To support the design of decision support systems, we desire to have a knowledge representation that supports modeling uncertainty and is flexible, intuitive, and mathematically sound. A Bayesian knowledge base (BKB) is a probabilistic knowledge representation meeting the preceding qualities [145]. A BKB supports theoretically sound and consistent probabilistic inference—even with incomplete knowledge—with the intuitiveness of "if-then" rule specification.[20]

Inherent in the BKB knowledge representation are several consistency constraints endowing the resulting knowledge base the ability to detect problems with the knowledge acquired and which can help alert the user to these possible problems [141]. Certain consistency constraint violations can be corrected without user intervention, with an appropriate status message displayed to the user. For others violations, user intervention is required. Users may correct the violation using one of the PESKI intelligent interface agents (e.g., knowledge acquisition, data mining, verification and validation).

PESKI provides users with engineering agents for knowledge acquisition [141], verification and validation [144, 142], data mining [155], and inferencing [149], each capable of operating in various communication modes to the user.

Using the active user interfaces paradigm, PESKI provides a complete and integrated suite of

[20]The representation is similar to Bayesian Networks [130]; it is a directed graph capable of representing uncertainty in knowledge via probabilistic relationships between random variables. However, Bayesian networks do not allow for incompleteness.

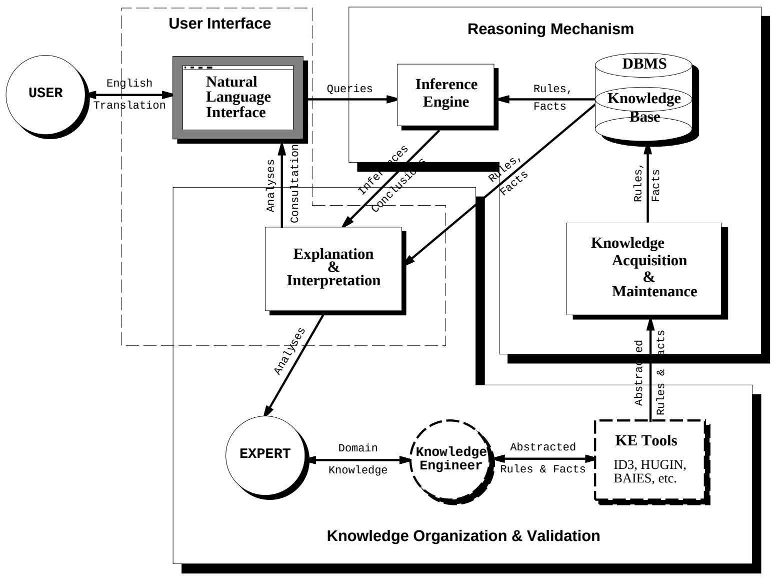

Figure 2: PESKI architecture.

knowledge engineering agents for constructing expert systems in nearly any domain (see Figure 2). In fact, other alternative architectures can be subsumed by the PESKI architecture [28].

4.4.1 Active User Interfaces

As we have been mentioning thus far, PESKI consists of a suite of engineering agents using the active user interface paradigm. We now briefly describe a subset of these agents as they are currently integrated into the PESKI architecture:

- **Knowledge Acquisition & Verification**—PESKI uses the MACK agent for knowledge acquisition [141]. MACK is designed to automatically and incrementally confirm consistency of the knowledge elicited from the expert and provides assistance by identifying the source of any inconsistency and proactively suggesting corrections. Regular incremental checks preserve both probabilistic validity and logical consistency as knowledge is acquired presumably under the expert's current consideration.

- **Validation**—PESKI validation is performed using two agents: BVAL [144] and GIT [142]. BVAL validates a knowledge base against its requirements using a test case-based approach. A test case is a set of evidence and expected answers. A knowledge engineer submits a test suite to the BVAL agent and BVAL determines if the given evidence is supported by the answers by submitting a query to the inference engine and comparing the solution with the test case's expected answer. Under certain conditions, the knowledge base is corrected automatically via reinforcement learning of the probabilities. For those test cases that indicate an incompleteness in the knowledge not meeting the test conditions (such as a missing causal relationship between two random variables), the graphical incompleteness tool (GIT) is used to visualize the knowledge base incompleteness for the user and actively provides solutions to correct it. GIT uses data visualization of the BKB and guides the user via color-coded shadings on how to repair the problem.

To recap, PESKI supports incremental knowledge elicitation in a number of ways. During knowledge acquisition, the user is alerted to any inconsistencies in the BKB knowledge representation. For example, if the user attempts to add a rule that creates a cycle in the knowledge base, PESKI will display an error message and attempt to provide assistance to the user in resolving this.

4.4.2 Active Intelligent Assistance

As we have described thus far, each agent in PESKI provides active assistance to the expert in the performance of various tasks. Still, these agents are somewhat myopic and focused on collaborating with the user to solve a particular task such as a validation failure for example. Determining which agent to use given a particular situation in PESKI can still be difficult for most users. The use of a particular agent is dependent on a number of variables including the context (e.g., a BKB constraint violation exists) and user preferences for the agents and various communication modes. Determining the correct agent to use at the correct time can be a daunting task. To aid users in efficiently utilizing the power of the PESKI agent suite offered, we have integrated an intelligent assistant into PESKI applying the active user interface paradigm at this highest level [74, 22, 20]. The assistant takes the form of an interface agent, "looking over the shoulder" of the user [27]. The overall goal of the assistant is to offer timely, beneficial assistance to the user as he/she interacts with PESKI. To accomplish this goal, an accurate cognitive model of the user is maintained [26]. The user model captures the goals and needs of the user within the PESKI environment, as well as possible system events that occur, within a probabilistic representation/model of the PESKI environment. Additionally, a user profile is maintained on each user of PESKI so assistance may be custom tailored to individual users. The interface agent determines the how, when, what, and why of offering assistance to the user by inferencing over the user model and utility functions [24]. The agent acts as a rational decision maker on behalf of the user, using the maximum expected utility principle of decision theory to choose the goal with the maximum expected utility and suggests that goal. The agent is capable of offering assistance for such goals as which agent to use to correct a BKB consistency constraint violation as well as suggesting the user preferred communication mode for a given agent. We are currently modifying the intelligent agent's architecture to allow it to collaboratively elicit information from the user based on what goals he/she is trying to achieve, his/her preferences, and past actions. To that end, we are adding "deep" domain knowledge of BKBs to the interface agent's user model. Additionally, we are developing an interface agent development environment to assist software developers implement interface agents into other domains.

4.5 In the Long Run ...

The previous two examples serve to show the benefits that can be gained in the area of user task load reduction via the use of the active user interface metaphor. Clavin highlights the autonomous, assistive, and adaptive characteristics of the active user interface, whereas PESKI highlights the assistive, auditable, and addressable characteristics of the metaphor. However, several of the characteristics are not realized in either of these two domains. Some are missing as a result of "sins of commision." That is, certain characteristics (e.g., adaptability) were purposely excluded from these domains because they were deemed unnecessary for these domains. Other characteristics are omitted, but should be included to improve the systems ability to aid the user in reduing his/her task load. For example, currently the active user interface possesses only rudimentary analytic capability to reason about its own actions.

In the (near) future, we envision an active user interface capable of exhibiting all the characteristics to a degree that best meets the needs of the user. Consider the following detailed scenario for Active User Interfaces:

> Bob, a system administrator, vigilantly monitors all highly sensitive company networks against the rising number of information warfare attacks that have recently been launched by competitors and rogue hackers. As Bob is watching the late-night news, a network alert is displayed by AUI[21] to Bob's terminal. The alert is in response to a very high level of network activity in links connected to the company's main supplier. AUI uses standard real-time graph visualization of the packet rate on the line. Bob requests AUI to "show" the alert, focusing attention from the news to the alert. Bob asks AUI to "show the network." AUI knows from the context that he is referring to the supplier's network and presents him with a graphical representation of the network topology. AUI focuses attention on a firewall that is the "target" of the increased packet rate. Bob asks "what type of architecture the firewall is running on?" A new window is displayed, showing the machine type information for the firewall. AUI accomplishes this by first consulting internal databases and then requesting information from the machine itself if necessary to expand its database. From this response, Bob learns the machine type. AUI anticipates Bob will want to know the operating system (OS) type associated with the machine and proactively displays the operating system. Bob is concerned about the vulnerabilities of this machine and OS type and asks AUI to display the vulnerabilities. With this request, AUI takes the combination of machine type and OS and conducts a search of all of its internal vulnerability databases as well as searching the most current resources on the web. It returns to Bob a list of all vulnerabilities that match the firewall machine type and OS. A number of fixes are identified with the vulnerabilities and in anticipation of Bob's request, AUI downloads, but does not install, the patches. Bob asks AUI if patches exist for these vulnerabilities and AUI returns with list of which have been patched and which have not. He then instructs AUI to retrieve and install the patches. Since AUI has prefetched the patches, it installs them immediately.
>
> Bob wants to get in touch with the system administrator of the firewall machine. He simply asks AUI who the system administrator is. AUI consults its own internal directories and returns the system administrator's name, John Doe, and his phone number. In previous interactions between Bob and AUI, AUI has returned all sorts of information about the individual Bob had inquired about. As a result, AUI has developed a detailed compendium of information about this system administrator. But Bob had asked AUI who the system administrator is. Using the "knowledge" of Bob's preferences from a larger context than the immediate one, AUI is able to discern that it should only display the name and phone number in this case and thereby intelligently filter out extraneous information that is irrelevant to the current context. Bob then asks AUI to place a video-call to the administrator. Bob tells John of the situation and sends a couple patches that John did not have for him to install.
>
> He asks AUI for the attacking machine information. Since he had asked for the machine type and the OS earlier, AUI presents both pieces of information to him. Bob then asks for denial of service (DOS) attacks against the hacker's machine/OS type combination. Unfortunately, because of the very recent dialogue with Bob concerning operating systems, AUI misinterprets Bob's request for information on "DOS" to refer to an operating system and returns some information on

[21]AUI is our imaginary/visionary active user interface for Bob.

the Diskette Operating System. Bob corrects AUI, informing AUI of this particular meaning of DOS. This collaborative interaction with AUI allows the system to correct its user model for future interactions. AUI replies with the correct information. Bob affirms the new information presented was what he is looking for and instructs AUI to target the attacking machine. AUI pops up a window showing the progress of the attack. The target is disabled and AUI installs a monitor to watch the target for future attacks. Bob can see the traffic in the network usage graph start to fall off and he goes back to watching the news to await for the next incident.

This scenario, while only imaginary in nature[22], demonstrates the full capabilities and ideas behind our active user interfaces metaphor. A number of issues are raised for future research in this scenario:

- *Credibility.* As previously mentioned, an advantage of agent-based systems is their ability to perform tasks on the user's behalf. As a result of this advantage, the question arises concerning how to build human confidence in the system's abilities to "do the right thing." [161] present an overview of credibility in computing technology. They contend credibility is synonymous with the term "believability." The phrases "trust in information," "trust in the output," and "accepting the advice" all refer to the credibility of the computer system.

  There are two extremes of the trust issue. At one end, the skeptic refuses to rely on data provided by these systems. [5] state one goal of human-computer interaction research is to ensure the user has a feeling of control. Users may feel they have lost control when they have no idea concerning what the system is doing nor what sort of processing the system performs to transform raw data into (possibly) useful information. [93] states "experts prefer to build their own mental models rather than rely on the aggregation and analyses of subordinates who are less skilled." This skepticism is certainly justified when the "subordinate" is a computer system. On the other end of the spectrum is "blind trust," or over reliance on the agent-based systems. Kilpatrick[23] states "as we train more and more reliance on computer systems ... we are not training the related common sense, and for lack of a better word, skepticism of computer data. [We] are growing [users] that will blindly trust computer-supplied data, which will make them very vulnerable ... " Interactive, visual user models can address the problem of lack of credibility. These user models allow the user to collaboratively help the system modify the user model and allow users to build their own mental models of the systems capabilities [19]. Since the visualization can include the system's capabilities and procedures to perform a task, these models provide the ability to train the user. Furthermore, collaborative (i.e., human-in-the-loop) and automated (e.g., machine learning) methods for improving the user model's performance by interactively correcting its knowledge can improve the user's confidence.

- *Information Pedigree.* For an active user interface to reason about building a knowledge plan for information visualization, it must also reason about the state of information held by other agents including the human user(s). This state may contain irrelevant, incorrect, dissonant, and/or missing information. Therefore, we are concerned with the information pedigree. That is, we desire to know the answers to question such as "who 'owns' the data/information?", "are they a trusted agent?", "how was the information derived?", "how current is the information?", etc. Obviously, information pedigree is related to the credibility issue discussed

[22]At least at the time of this writing.
[23]Alex Kilpatrick, Air Force Office of Scientific Research, personal communication.

previously. Methods to visualize this entire process of information pedigree development are needed and are paramount to increasing user trust.

- *Information Dynamics.* It is anticipated that new information in any nontrivial domain of interest may arrive asynchronously, and thus AUI must continually update this information. That is, one cannot assume complete and consistent information *ab initio*. Instead, a knowledge source is only partially complete at any point in time and may contain substantial noise. If new information demands further elaboration, the execution of information-gathering actions may be required to produce additional details. Therefore, information may be presented to the user before complete details are known. The user(s) needs to be able to visualize how the information "unfolds" over time. Although the ideas are extensions and adaptations of existing results in the machine-planning and learning literature, the application of such results to human/information interaction is novel. Results from [43] provides a foundation for building a theory of errors in information interaction and recovery from it. Further research reported in [163] and [44] contribute a foundational theory of rational-driven sensing monitors that link changes in the environment to the adaptation of plans and goals in continuous planning environments. This foundation applies generally, despite the class of planner being used (e.g., state-space or partial order) or the domain being manipulated. Mapping to an information domain will require making the necessary interpolations to algorithms and appropriate development of representations. Note, however, that current research has already demonstrated the fundamental relationship between planning and information acquisition [43, 52, 137].

- *Data Aggregation.* The problem of data aggregation is one of finding evidence of an event's occurrence among data from multiple sources. For information warfare[24] a further requirement is that the functional capabilities being targeted will be identified. This is an evidence accrual challenge because the attacks may be distributed over a range of means, points of origin, and temporal/spatial occurrence. Assessment is required to support defensive information warfare situational awareness and defensive response selection, such that the data and hypothesized patterns are made available to the decision-maker at an appropriate level of cognitive abstraction. The data aggregation problem can be specified with formal declarative statements, starting with a collection of formally stated facts directly describing data patterns at their distributed collection points. In the case of computer networks the data could be the network activity logs at each node. As facts from different network nodes are aggregated, patterns can be matched against aggregations. Candidate patterns include typical hacker activity to gain control of the network, or the initial effects of a recently inserted virus. As the process is repeated greater amounts of data are aggregated and matched against successively more complex patterns. Larger aggregations are represented by more detailed specifications which are consistent with the fewer specifications associated with earlier (smaller) aggregations. Consistency across iterations of this process indicates computationally tractable specification refinement. (Consistency is guaranteed because at any iteration the aggregated data is ultimately traceable back to the network nodes where data was first gathered.) Visualization, as pertinent to the situation assessment/response selection aspects of the defensive information warfare problem goes beyond the conventional scientific data visualization techniques.

[24]While this discussion assumes the information warfare scenario, the data aggregation problem is not limited to this domain.

A cognitively-based schema is sought which intuitively depicts the spatial, temporal, hierarchical, geographic, functional, and/or informational organizations of incident data. It is recognized that these data will be aggregated over both time of occurrence and location.

- *User-Based Relevance.* Relevance plays a key role in the evaluation of information retrieval systems. In recent years, a number of researchers have realized that existing topical relevance-based systems have fallen short of meeting real users' needs [58, 128, 12]. User-based relevance takes into account an individual's subjective contexts and personal needs as an underlying situational force in information seeking and retrieval processes [128]. [58] notes several common themes that arise in information retrieval research:

  1. the inability to define relevance;
  2. the inadequacy of topicality as the basis for relevance judgments;
  3. the diversity of non-topical, user-centered criteria that affect relevance judgments;
  4. the dynamic and fluid character of information seeking behavior;
  5. the need for appropriate methodologies; and
  6. the need for more complex, robust models for system design and evaluation.

  To be sure, the incorporation of user-based relevance measures with the "prototypical core" of relevancy (i.e., topicality) [58] can strengthen an information systems ability to find relevant information for users.

5 Summary

The voluminous amount of information available to users to perform their tasks and the complexity of existing interfaces imposes an unmanageable cognitive burden upon the user. The problem of increasing user task load resulting from information overload and under load has been, to date, addressed by interface agents and user models.

Interface agents extract and analyze relevant information, providing information abstractions of that information, and providing timely, beneficial Even though the agent community can not agree upon a definition for the term "agent", they have proven to be a useful paradigm for system design in many domains. The main interface agents metaphors, assistant and associate, communicate with the user through the existing user interface. The interface to the user of these agents range from highly anthropomorphized interfaces to simple, but powerful direct manipluation interfaces.

More than any other type of agent, interface agents must be able to effectively and efficiently interact with the user. To accomplish this goal, interface agents use some sort of user model to represent users' knowledge and interaction within a system. These user models allow the system to adapt to the needs of users. Human-computer interaction and artificial intelligence have both greatly influenced user modeling research. Both fields have influenced the various methodologies for elicitation and specification, design, and implementation of interface agents. The inclusion of a user model within the overall system architecture allows the system to adapt its response to the preferences, biases, expertise level, goals and needs.

Active user interfaces are the next logical step towards addressing the information overload problem. The new user interface metaphor, that of an actuator to the human-computer interface, is the result of a symbiotic relationship between best practices from the human-computer interaction, artificial intelligence and user modeling communities. These active user interfaces are capable of multi-levels of collaboration and autonomy. The user of an active user interface is fully aware of any interface actions taken by the active user interface and has a complete, intuitive understanding of such actions. This insures the user is always in control of the interface's actions. We believe the use of the evolutionary active user interface metaphor for human-system interaction will bring about further gains in reducing the user's task load. We have provided two examples of first-generation active user interfaces to illustrate this belief. These two systems—Clavin and PESKI—are situated within the domains of a natural language query information management system and an expert system shell and show how the active user interface is capable of providing an intuitive, meaningful, and useful "actuation" interface for a heterogeneous collection of information sources and tools. These two examples explicitly highlight several key characteristics of the active user interface metaphor. At the same time, they implicitly highlight characteristics these interfaces do not yet possess. Furure work in these two domains and others will serve to continue to evolve this new metaphor, showing its strengths and highlighting areas that need further evolution.

References

[1] Irina Akoulchina and Jean-Gabriel Ganascia. SATELIT-Agent: An adaptive interface based on learning interface agents technology. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 21–32. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[2] David W. Albrecht, Ingrid Zukerman, Ann E. Nicholson, and Ariel Bud. Towards a Bayesian model for keyhole plan recognition in large domains. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 365–376. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[3] Leonardo Ambrosini, Vincenzo Cirillo, and Alessandro Micarelli. A hybrid architecture for user-adapted information filtering on the World Wide Web. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 59–61. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[4] Elisabeth André, Thomas Rist, and Jochen Müller. WebPersona: a lifelike presentation agent for the World-Wide Web. *Knowledge-Based Systems*, 11:25–36, 1998.

[5] Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton, and Saul Greenberg. From customizable systems to intelligent agents. In *Readings in Human-Computer Interaction: Toward the Year 2000*, chapter 12, pages 783–792. Morgan Kaufmann, second edition, 1995.

[6] Sheila B. Banks, Robert A. Harrington, Eugene Santos Jr., and Scott M. Brown. Usability testing of an intelligent interface agent. In *Proceedings of the Sixth International Interfaces Conference (Interfaces 97)*, pages 121–123, May 1997.

[7] Sheila B. Banks and Carl S. Lizza. Pilot's Associate: A cooperative, knowledge-based system application. *IEEE Expert*, pages 18–29, June 1991.

[8] Sheila B. Banks, Martin R. Stytz, Eugene Santos Jr., and Scott M. Brown. User modeling for military training: Intelligent interface agents. In *Proceedings of the 19th Interservice/Industry Training Systems and Education Conference*, pages 645–653, December 1997.

[9] K. S. Barber and C. E. Martin. Agent autonomy: Specification, measurement, and dynamic adjustment. In *Proceedings of Agent Autonomy Workshop*, pages 8–15, May 1999. held in conjunction with the Third Autonomous Agents Conference (Agents '99), Seattle, WA, 1–5 May 1999.

[10] Mihai Barbuceanu and Mark S. Fox. The design of a coordination language for multi-agent systems. In *Proceedings of the Agent Theories, Architectures, and Languages Conference*, pages 341–355, 1996.

[11] Don Barker. Secret agent man: IBM turns over a new leaf with Ginkgo. *PC AI*, 12(2):21–22, March/April 1998.

[12] Carol L. Barry. User-defined relevance criteria: An exploratory study. *Journal of the American Society for Information Science*, pages 135–141, April 1994.

[13] Mathias Bauer. Acquisition of user preferences for plan recognition. In Anthony Jameson, Cécil Paris, and Carlo Tasso, editors, *Proceedings of the Fifth Internataional Conference on User Modeling (UM '96)*, pages 105–112. SpringerWien, New York, 1996.

[14] D. Benyon and D. Murray. Adaptive systems: from intelligent tutoring to autonomous agents. *Knowledge-Based Systems*, 6(4):197–219, December 1993.

[15] Andre Berthold and Anthony Jameson. Interpreting symptoms of cognitive load in speech input. In *User Modeling: Proceedings of the Seventh International Conference, UM99*, June 1999.

[16] M. Billinghurst and J. Savage. Adding intelligence to the interface. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pages 168–176. Piscataway, NJ: IEEE Press, 1996.

[17] Kurt D. Bollacker, Steve Lawrence, and C. Lee Giles. CiteSeer: An autonomous web agent for autonomous retrieval and indentification of interesting publication. In Katia P. Sycara and Michael Woolridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 116–123. ACM Press, May 9–13 1998.

[18] Gary Boone. Concept features in Re:Agent, an intelligent email agent. In Katia P. Sycara and Michael Woolridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 141–148. ACM Press, May 9–13 1998.

[19] Scott M. Brown. *A Decision Theoretic-Based Approach for Interface Agent Development.* PhD thesis, United States Air Force Institute of Technology, 1998.

[20] Scott M. Brown, Robert A. Harrington, Eugene Santos Jr., and Sheila B. Banks. User models, interface agents and expert systems. In *Proceedings of the Embedding User Models in Intelligent Applications Workshop*, pages 12–17, June 1997. held in conjunction with the Sixth International Conference on User Modeling (UM '97).

[21] Scott M. Brown and Eugene Santos Jr. Intelligent interfaces for decision-theoretic systems. In *PESKI Invited Demonstration*, pages 41–42, 1999. held in conjunction with the Third International Conference on Autonomous Agents (Agents '99), Seattle, WA, 1–5 May 1999.

[22] Scott M. Brown, Eugene Santos Jr., and Sheila B. Banks. A dynamic Bayesian intelligent interface agent. In *Proceedings of the Sixth International Interfaces Conference (Interfaces 97)*, pages 118–120, May 1997.

[23] Scott M. Brown, Eugene Santos Jr., and Sheila B. Banks. Supporting incremental knowledge elicitation in decision-theoretic systems. In *Proecedings of the 1998 AAAI Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic Systems*, pages 14–15, March 1998.

[24] Scott M. Brown, Eugene Santos Jr., and Sheila B. Banks. Utility theory-based user models for intelligent interface agents. In *Proceedings of the Twelfth Biennial Conference of the Canadian Society for Computational Studies on Intelligence*, pages 378–392, June 1998.

[25] Scott M. Brown, Eugene Santos Jr., and Sheila B. Banks. Active user interfaces for building decision-theoretic systems. In *Proceedings of the First Asian-Pacific Conference on Intelligent Agent Technology*, 14–17 Dec 1999. to appear.

[26] Scott M. Brown, Eugene Santos Jr., Sheila B. Banks, and Mark E. Oxley. Using explicit requirements and metrics for interface agent user model correction. In Katia P. Sycara and Michael Woolridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 1–7. ACM Press, May 9–13 1998.

[27] Scott M. Brown, Eugene Santos Jr., Sheila B. Banks, and Martin R. Stytz. Intelligent interface agents for intelligent environments. In *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments*. AAAI Press, AAAI Technical Report SS-98-02, March 1998.

[28] Bruce G. Buchanan and David C. Wilkins, editors. *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*. Morgan Kaufmann, 1993.

[29] Susan Bull. See Yourself Write: A simple student model to make students think. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 315–326. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[30] Amedeo Cesta and Daniela D'Aloisi. Building interfaces as personal agents: a case study. *SIGCHI Bulletin*, 28(3):108–113, 1996.

[31] Amedeo Cesta and Daniela D'Aloisi. Mixed-initiative aspects in an agent-based meeting scheduler. *User Modeling and User-Adapted Interaction, Special issue on "Computational Models of Mixed-Initiative Interaction"*, 1999. To appear.

[32] Amedeo Cesta, Daniela D'Aloisi, and Vittorio Giannini. Active interfaces for software tools. In Y. Anzai, K. Ogawa, and H. Mori, editors, *Proceedings of the 6th International Conference on Human-Computer Interaction*, pages 225–230. Elsevier Science B.V., 1995.

[33] Liren Chen and Katia Sycara. WebMate: A personal agent for browsing and searching. In Katia P. Sycara and Michael Woolridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 132–140. ACM Press, May 9–13 1998.

[34] D. N. Chin. Intelligent interfaces as agents. In J. W. Sullivan and S. W. Tyler, editors, *Intelligent User Interfaces*. ACM, New York, 1991.

[35] Bark Cheung Chiu, Geoffrey I. Webb, and Mark Kuzmycz. A comparison of first-order and zeroth-order induction for Input-Output Agent Modelling. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 347–358. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[36] Philip R. Cohen. The role of natural language in a multimodal interface. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 143–149, 1992.

[37] Kevin Cole, Olivier Fischer, and Phyllis Saltzman. Just-in-time knowledge delivery. *Communications of the ACM*, 40(7):49–53, July 1997.

[38] Cristina Conati, Abigail S. Gertner, Kurt VanLehn, and Marek J. Druzdzel. On-line student modeling for coached problem solving using Bayesian networks. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 231–242. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[39] Nancy J. Cooke. Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies*, 41(6):801–849, 1994.

[40] Alan Cooper. The myth of metaphor. Available at http://www.cooper.com/articles/vbpj_myth_of_metaphor.htm, 1995.

[41] Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[42] Albert T. Corbett and Akshat Bhatnagar. Student modeling in the ACT programming tutor: Adjusting a procedural learning model with declarative knowledge. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 243–254. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[43] M. Cox. *Introspective multistrategy learning: Constructing a learning strategy under reasoning failure.* PhD thesis, Georgia Institute of Technology, College of Computing, Atlanta, 1996.

[44] M. T. Cox and G. Rasul. Human interaction with automated planners through the manipulation and visualization of goal change. In *Proceedings of the AAAI-99 Workshop on Mixed-Initiative Intelligence*, 1999.

[45] Andrew Csinger, Kellogg S. Booth, and David Poole. AI meets authoring: User models for intelligent multimedia. *Artificial Intelligence Review*, 8:447–468, 1994.

[46] Andrew Csinger and David Poole. User models and perceptual salience: Formal abduction for model recognition and presentation design. In *User Modeling: Proceedings of the Fifth International Conference, UM96*, pages 51–58, 1996.

[47] Allen Cypher. Eager: Programming repetitive tasks by example. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 33–39, 1991.

[48] K. S. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In W. L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 404–413, Marina del Rey, February 1997.

[49] Richard DeWitt. Vagueness, semantics, and the language of thought, 1995.

[50] Anne-Claude Doux, Jean-Philippe Laurent, and Jean-Pierre Nadal. Symbolic data analysis with the K-means algorithm for user profiling. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference,*

UM97, pages 359–361. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[51] Clark Elliot. Hunting for the holy grail with “emotionally intelligent” virtual actors. *SIGART Bulletin*, 9(1):20–28, Summer 1998.

[52] O. Etzioni and D. Weld. A softbot-based interface to the internet. *Communications of the ACM*, 37(7):72–76, 1994.

[53] Andrew E. Fano. SHOPPER'S EYE: Using location-based filtering for a shopping agent in the physical world. In Katia P. Sycara and Michael Woolridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 416–421. ACM Press, May 1998.

[54] Josef Fink, Alfred Kobsa, and Andreas Nill. Adaptable and adaptive information access for all users, including the disabled and the elderly. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 171–173. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[55] John Flach and Gil Kuperman. Vicory by design: War, information, and cognitive systems engineering. Technical Report AFRL-HE-WP-TR-1998-0074, United States Air Force Research Laboratory, February 1998.

[56] Leonard Newton Foner. Paying attention to what's important: Using focus of attention to imporve unsupervised learning. Master's thesis, Massachusetts Institute of Technology, June 1994.

[57] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag, 1996.

[58] Thomas J. Froehlich. Relevance reconsiderd—towards an agenda for the 21st century: Introduction to special topic issue on relevance research. *Journal of the American Society for Information Science*, pages 124–133, April 1994.

[59] Tatjana Gavrilova and Alexander Voinov. An approach to mapping of user model to corresponding interface parameters. In *Proceedings of the Embedding User Models in Intelligent Applications Workshop*, pages 24–29, June 1997. held in conjunction with the Sixth International Conference on User Modeling (UM '97).

[60] Abigail S. Gertner, Cristina Conati, and Kurt VanLehn. Procedural help in Andes: Generating hints using a Bayesian network student model. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, San Francisco, CA, 1998. Morgan Kaufmann Publishers.

[61] Paolo Giangrandi and Carlo Tasso. Managing temporal knowledge in student modeling. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 415–426. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[62] Robert P. Goldman and Eugene Charniak. A language for construction of belief networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):196–208, 1993.

[63] Avelino J. Gonzalez and Douglas D. Dankel. *The Engineering of Knowledge-Based Systems: Theory and Practice.* Prentice-Hall, Inc., 1993.

[64] Richard Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie Mellon Institute, 1993.

[65] Marco Gori, Marco Maggini, and Enrico Martinelli. Web-browser access through voice input and page interest prediction. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 17–19. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[66] Mark Green and Robert Jacob. Software architectures and metaphors for non-wimp user interfaces. Notes from ACM SIGGRAPH'90, February 1991.

[67] Bernd Gutkauf, Stefanie Thies, and Gitta Domik. A user-adaptive chart editing system based on user modeling and critiquing. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 159–170. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[68] Vu Ha and Peter Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–97)*, pages 215–222, Providence, Rhode Island, 1997.

[69] Susan Haller and Susan McRoy. Special issue: Computational models of mixed-initiative interaction (part i). *User Modeling and User-Adapted Interaction*, 8(3-4), 1998.

[70] P. Hamil. Psychological issues in the design of expert systems. In *Proceedings of the Human Factors Society—28th Annual Meeting*, 1984.

[71] John M. Hammer and Ronald L. Small. An intelligent interface in an associate system. *Human/Technology Interaction in Complex Systems*, 7:1–44, 1995.

[72] Kristian J. Hammond. Anticipating user's needs: Redeeming big brother in the information age. In Mark Maybury, editor, *Proceedings of the International Conference on Intelligent User Interfaces*, pages 181–182, Redondo Beach, Los Angeles, CA, January 1999. Plenary Address.

[73] Eui-Hong (Sam) Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher, and Jerome Moore. WebACE: A web agent for document categorization and exploration. In Katia P. Sycara and Michael Woolridge, editors, *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*, pages 408–415. ACM Press, May 1998.

[74] Robert A. Harrington, Sheila Banks, and Eugene Santos Jr. GESIA: Uncertainty-based reasoning for a generic expert system intelligent user interface. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*, pages 52–55, 1996.

[75] Robert A. Harrington and Scott M. Brown. Intelligent interface learning with uncertainty. In Eugene Santos Jr., editor, *Proceedings of the Eighth Midwest Artificial Intelligence and Cognitive Science Conference*, pages 27–34. AAAI Press, 1997.

[76] Graeme Hirst, Chrysanne DiMarco, Eduard Hovy, and Kimberley Parsons. Authoring and generating health-education documents that are tailored to the needs of the individual patient. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 107–118. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[77] Kristina Höök. Steps to take before IUI becomes real. Presented at the Workshop "The Reality Of Intelligent Interface Technology", March 1997.

[78] Eric Horvitz. Agents with beliefs: Reflections on Bayesian methods for user modeling. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 441–442. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[79] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 159–166, May 1999.

[80] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. The Lumiére project: Bayesian user modeling for inferring goals and needs of software users. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, San Francisco, CA, 1998. Morgan Kaufmann Publishers.

[81] IBM Corp. IBM intelligent agents Ginkgo white paper. World Wide Web Page http://www.networking.ibm.com/iag/iaginkgo.htm, 1998.

[82] Anthony Jameson. Numeric uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interactions*, 5:193–251, 1996.

[83] Anthony Jameson, Cécil Paris, and Carlo Tasso, editors. *Preface of User Modeling: Proceedings of the Sixth International Conference, UM97.* Springer Wien New York, 1997.

[84] Pertti Järvnin. Notes on assumptions of user modelling. Technical Report A-1993-2, University of Tampere, Department of Computer Science, P.O. Box 607, SF-33101 Tampere, Finland, March 1993.

[85] N. R. Jennings, Katia Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[86] W. Lewis Johnson and Jeff Rickel. Steve: An animated pedagogical agent for procedural training in virtual environments. *SIGART Bulletin*, 8(1–4):16–21, Fall 1998.

[87] Slava Kalyuga, Paul Chandler, and John Sweller. Levels of expertise and user-adapted formats of instructional presentations: A cognitive load approach. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference,*

UM97, pages 261–272. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[88] Charalampos Karagiannidis, Adamatios Koumpis, and Constantine Stephanidis. Deciding ‘what’, ‘when’, ‘why’, and ‘how’ to adapt in intelligent multimedia presentation systems. In G.P. Faconti and T. Rist, editors, *Proceedings of the Twelfth European Conference on Artificial Intelligence Workshop "Towards a Standard Reference Model for Intelligent Multimedia Presentation Systems"*. John Wiley & Sons, Ltd., August 1996.

[89] Judy Kay. Lies, damn lies, and stereotypes: pragmatic approximations of users. In *User Modeling: Proceedings of the Fourth International Conference, UM94*, pages 175–184. 1994. Early draft of invited keynote presented at UM’94 available at http://www.cs.su.oz.au/ judy/Research/index.html.

[90] Simeon Keates and Peter Robinson. User performance modeling and cognitive load. In *Proceedings of the RESNA Annual Conference*, pages 342–344. RESNA Press, June 1997.

[91] William Joseph King and Jun Ohya. The representation of agents: Anthropomorphism, agency, and intelligence. In *Electronic Proceedings of CHI96*, 1996.

[92] Jak Kirman, Ann Nicholson, Moises Lejter, Thomas Dean, and Eugene Santos Jr. Using goals to find plans with high expected utility. In *Proceedings of the Second European Workshop on Planning*, pages 158–170, Linkoping, Sweden, 1993.

[93] G. Klein. Implications of the naturalistic decision making framework for information dominance. Technical Report AL/CF-TR-1997-0155, United States Air Force Armstrong Laboratory, July 1997.

[94] Tomoko Koda and Pattie Maes. Agents with faces: The effects of personification of agents. In *Proceedings of HCI’96*, 1996.

[95] Robyn Kozierok and Pattie Maes. A learning interface agent for scheduling meetings. In *Proceedings of the ACM SIGCHI International Workshop on Intelligent User Interfaces*, pages 81–88, 1993.

[96] Yezdi Lashkari, Max Metral, and Pattie Maes. Collaborative interface agents. In *Proceedings of the National Conference on Artificial Intelligence*, Cambride, MA, 1994. MIT Press.

[97] Steve Lawrence, C. Lee Guiles, and Kurt D. Bollacker. Autonomous citation matching. In Oren Etzioni, Jörg P. Müller, and Jeffrey Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents ’99)*, pages 392–393. ACM Press, May 1–5 1999.

[98] Michael Lewis. Designing for human-agent interaction. *AI Magazine*, 19(2):67–78, 1998.

[99] Henry Lieberman. A user interface for knowledge acquisition from video. In Allen Cypher, editor, *Conference of the American Association for Artificial Intelligence*. MIT Press, 1994.

[100] Henry Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[101] Henry Lieberman. Autonomous interface agents. In *Proceedings of the Computer-Human Interaction (CHI) Conference*, pages 67–74. ACM Press, 1997.

[102] Hnery Lieberman. Integrating user interface agents with conventional applications. *Knowledge-Based Systems*, 11:15–23, 1998.

[103] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The Automated Travel Assistant. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 67–78. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[104] Patti Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1 & 2), 1994. MIT Press (C. Langton, Ed.).

[105] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):811–821, July 1994.

[106] Pattie Maes and Robyn Kozierok. Learning interface agents. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), Washington, DC*, 1993.

[107] Pattie Maes and Benjamin Schneiderman. Direct manipulation vs. interface agents: A debate. *Interactions*, 4(6), 1997.

[108] Paul P. Maglio and Rob Barrett. How to build modeling agents to support web searchers. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 5–16. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[109] T. Malone, K.Y. Lai, R. Rao, and D. Rosenblitt. The Information Lens: An intelligent system for information sharing and coordination. In M. Olson, editor, *Tecnological Support for Working Group Collaboration*. Lawrence Erlbaum Associates, 1989.

[110] Amir Mane, Susan Boyce, Demetrios Karis, and Nicole Yankelovich. Designing the user interface for speech recognition applications, A CHI96 Workshop. *SIGCHI Bulletin*, 28(4), 1996.

[111] C. E. Martin and K. S. Barber. Representation of autonomy in distributed agent-based systems. In *Intelligent Systems: A Semiotic Perspective. Proceedings of the 1996 International Multidisiplinary Conference*, pages 67–72, October 1996.

[112] C. E. Martin, R. H. Macfadzean, and K. S. Barber. Supporting dynamic adaptive autonomy for agent-based systems. In *Proceedings of 1996 Artificial Intelligence and Manufacturing Research Planning Workshop*, pages 112–120, June 1996.

[113] P. Meseguer and A. Verdaguer. Verification of multi-level rule-based expert systems: Theory and practice. *International Journal of Expert Systems*, 6:163–192, 1993.

[114] Max Edward Metral. Design of a generic learning interface agent. Bachelor of Science, Massachussets Institute of Technology, May 1993.

[115] Christopher Miller and Raymond Larson. An explanatory and "argumentative" interface for a model-based diagnostic system. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 43–52, 1992.

[116] Christopher A. Miller and Matthew D. Hannen. User acceptance of an intelligent user interface: A rotorcraft pilot's associate example. In Mark Maybury, editor, *Proceedings of the International Conference on intelligent User Interfaces*, pages 109–119, Redondo Beach, Los Angeles, CA, January 1999.

[117] Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, July 1994.

[118] Sandeep S. Mulgund and Greg L. Zacharias. A situation-driven adaptive pilot/vehicle interface. In *Proceedings of the Third Annual Symposium on Human Intercation with Complex Systems*, pages 193–198, August 1996.

[119] Jörg P. Müller. A cooperation model for autonomous agents. In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Intelligent Agents III: Agent Theories, Architectures, and Languages*, ECAI'96 Workshop (ATAL), pages 245–260. Springer, August 1996.

[120] Maureen Murphy and Michael McTear. Learner modelling for intelligent CALL. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 301–312. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[121] Brad A. Myers. A brief history of human computer interaction technology. Technical Report CMU-CS-96-163, Carnegi Mellon University, December 1996.

[122] T.A. Nguyen. Verifying consistency of production systems. *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, pages 4–8, February 1987.

[123] Sanguk Noh and Piotr J. Gmytrasiewicz. Agent modeling in antiair defense: A case study. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 389–400. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[124] Donald A. Norman. How might people interact with agents. *Communications of the ACM*, pages 68–71, July 1994.

[125] Celestine A. Ntuen. A formal method for deriving command production language from human intents. In *Proceedings of the Seventh International Conference on Human-Computer Interaction (HCI International '97)*, August 1997.

[126] R. M. O'Keefe, O. Balci, and E. P. Smith. Validating expert system performance. *IEEE Expert*, 2(4):81–90, 1987.

[127] Priyanka Paranagama, Frada Burstein, and David Arnott. Modelling the personality of decision makers for active decision support. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 79–81. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[128] Taemin Kim Park. Toward a theory of user-based relevance: A call for a new paradigm of inquiry. *Journal of the American Society for Information Science*, pages 135–141, April 1994.

[129] Simon Parsons. Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 8:353–372, June 1996.

[130] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[131] Charles Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, 11(6):24–29, December 1996.

[132] Wolfgang Pohl. LaboUr – machine learning for user modeling. In *Proceedings of the Seventh International Conference on Human-Computer Interaction (HCI International '97)*, pages 27–30, August 1997.

[133] Wolfgang Pohl and Jörg Höhle. Mechanisms for flexible representation and use of knowledge in user modeling shell systems. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 403–414. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[134] David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[135] G. Ravi Prakash and H.N. Mahabala. Svepoa: A tool to aid verification and validation of ops5-based ai applications. *International Journal of Expert Systems*, 6:193–236, 1993.

[136] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[137] A. Ram and L. Hunter. The use of explicit goals for knowledge to guide inference and learning. *Applied Intelligence*, 2:47–73, 1992.

[138] Bradley J. Rhodes. Remembrance agent: A continuously running automated information retrieval system. In *Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology, London, UK*, pages 487–495, April 1996.

[139] Elaine Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, 18:199–214, 1983.

[140] Frederick W. Rook and Michael L. Donnell. Human cognition and the expert system inetrface: Mental models and inference explanation. *IEEE Transactions on System, Man, and Cybernetics*, 23(6), 1993.

[141] Eugene Santos Jr., Darwyn O. Banks, and Sheila B. Banks. MACK: A tool for acquiring consistent knowledge under uncertainty. In *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, pages 23–32, 1997.

[142] Eugene Santos Jr., Sheila B. Banks, Scott M. Brown, and David J. Bawcom. Identifying and handling structural incompleteness for validation of probabilistic knowledge-bases. In *Proceedings of the 11th International FLAIRS Conference*, pages 506–510, 1999.

[143] Eugene Santos Jr., Scott M. Brown, Moises Lejter, Grace Ngai, Sheila B. Banks, and Martin R. Stytz. Dynamic user model construction with Bayesian networks for intelligent information queries. In *Proceedings of the 11th International FLAIRS Conference*, pages 3–7, May 1999.

[144] Eugene Santos Jr., Howard T. Gleason, and Sheila B. Banks. BVAL: Probabilistic knowledge-base validation. In *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, pages 13–22, 1997.

[145] Eugene Santos Jr. and Eugene S. Santos. A framework for building knowledge-bases under uncertainty. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:265–286, 1999.

[146] Tefko Saracevic, Amanda Spink, and Mei-Mei Wu. Users and intermediaries in information retrieval: What are they talking about? In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 43–54. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[147] Ralph Schäfer and Thomas Weyrath. Assessing temporally variable user properties with dynamic Bayesian networks. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 377–388. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[148] Ted Selker. COACH: A teaching agent that learns. *Communications of the ACM*, 37(7):92–99, July 1994.

[149] Solomon Eyal Shimony, Carmel Domshlak, and Eugene Santos Jr. Cost-sharing heuristic for Bayesian knowledge-bases. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 421–428, 1997.

[150] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, March 1993.

[151] Richard Stallman. *GNU Emacs Manual, Thirteenth Edition, Updated for Emacs Version 20.1.* Free Software Foundation, July 1997. ISBN 1-882114-06-X.

[152] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, December 1986.

[153] Chris Stary. The role of interaction modeling in future cognitive ergonomics: Do interaction models lead to formal specifications of involved machine intelligence? In *Proceedings of the Seventh International Conference on Human-Computer Interaction (HCI International '97)*, August 1997.

[154] Vadim L. Stefanuk. Embedding user models in intelligent interfaces. In *Proceedings of the Embedding User Models in Intelligent Applications Workshop*, pages 18–23, June 1997. held in conjunction with the Sixth International Conference on User Modeling (UM '97).

[155] Daniel J. Stein, Sheila B. Banks, Eugene Santos, Jr., and Michael Talbert. Utilizing goal-directed data mining for incompleteness repair in knowledge bases. In *Proceedings of the Eighth Midwest AI and Cognitive Science Conference*, pages 82–85, 1997.

[156] Oliviero Stock. Natural language in multimodal human-computer interfaces. *IEEE Expert*, pages 40–44, April 1994.

[157] Katia Sycara, Keith Decker, Anandeep Pannu, Mike Williamson, and Dajun Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, December 1996.

[158] Loren Terveen, Will Hill, and Brian Amento. Collaborative filtering to locate, comprehend, and organize collections of web sites. *SIGART Bulletin*, 3 & 4(9):10–17, 1998.

[159] Loren G. Terveen. Overview of human-computer collaboration. *Knowledge-Based Systems*, 8(2–3):67–81, April–June 1995.

[160] C. G. Thomas. Design, implementation and evaluation of an adaptive user interface. *Knowledge-Based Systems*, 6(4):230–238, December 1993.

[161] Shawn Tseng and B. J. Fogg. Credibility and computing technology. *Communications of the ACM*, 42(5):39–44, May 1999.

[162] Julita Vassileva. A new view of interactive human-computer environments. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 433–435. Springer Wien New York, Vienna, New York, 1997. Available from http://um.org.

[163] M. M. Veloso, M. E. Pollack, and M. T. Cox. Rationale-based monitoring for continuous planning in dynamic environments. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 171–179, 1998.

[164] Annika Waern. *Recognising Human Plans: Issues for Plan Recognition in Human-Computer Interaction.* PhD thesis, Royal Institute of Technology, 1996.

[165] Annika Waern. What is an intelligent interface? notes from an introduction seminar, March 1997.

[166] Patrick Henry Winston. *Artificial Intelligence, 2d Ed.* Addison-Wesley, 1984.

[167] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995. Available at ftp.elec.qmw.ac.uk/pub/isag/distributed-ai/publications/KE-REVIEW-95.ps.Z.

[168] Fahri Yetim. User-adapted hypertext explanations. In Thomas Grechenig and Manfred Tscheligi, editors, *Proceedings of the Vienna Conference Human Computer Interaction*, Lecture Notes in Computer Science, pages 91–102, Vienna, Austria, September 1993. Springer-Verlag.

[169] Kenichi Yoshida. User modeling by graph-based induction. In *Proceedings of the Seventh International Conference on Human-Computer Interaction (HCI International '97)*, pages 23–26, August 1997.

[170] Kenichi Yoshida and Hiroshi Motoda. Automated user modeling for intelligent interface. *International Journal of Human-Computer Interaction*, 8(3):237–258, 1997.

[171] David Zeng and Katia Sycara. How can an agent learn to negotiate? In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Intelligent Agents III: Agent Theories, Architectures, and Languages*, ECAI'96 Workshop (ATAL), pages 233–244. Springer, August 1996.