AN ABSTRACT OF THE THESIS OF

Shikha Ghosh Gottfried for the degree of Master of Science in Computer Science presented on December 5, 1996. Title: A Conceptual Framework for Web-based Collaborative Design.

Abstract a

Bruce D'Ambrosio

Although much effort has been invested to build applications that support group work, collaborative applications have not found easy success. The cost of adopting and maintaining collaborative applications has prevented their widespread use, especially among small distributed groups. Application developers have had difficulties recognizing the extra effort required by groups to use collaborative applications and how to either reduce this effort or provide other benefits to compensate for the extra work. These problems have limited the success of collaborative applications, which have not attained the same level of productivity improvements that single user applications have achieved. In this thesis we present a framework that describes the types of computer support that can facilitate the work of distributed engineering design groups. Our framework addresses support for web-based groups in particular because we believe the web can be a powerful medium for collaboration if accommodated properly. We show how the concepts in this framework can be implemented by prototyping a web-based engineering decision support system. Our framework is a synthesis of ideas motivated by an examination of literature in various fields that share a common interest in collaborative work. It can influence application development by helping developers become aware of the types of support should be considered to aid web-based collaborative design.

A Conceptual Framework for Web-based Collaborative Design

by

Shikha Ghosh Gottfried

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed December 5, 1996
Commencement June 1997

Master of Science thesis of Shikha Ghosh Gottfried presented on December 5, 1996

APPROVED:

Major Professor, representing Computer Science

Chair of Department of Computer Science

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for privacy

Shikha Ghosh Gottfried, Author

TABLE OF CONTENTS

List of Figures

Figure Page

A Conceptual Framework For Web-based Collaborative Design

1. Introduction

Research of the design of computer systems that support collaboration has been fueled by advances in technologies that enable shared work as well as a continuing need to find solutions that effectively aid projects that require group effort. Groups are no longer bound by distance or time; users are accustomed to communicating with others who can be anywhere in the world, exchanging information at different times. The ease with which such communication is possible has made users eager to extend simple communication to more sophisticated group interaction. Despite the interest in this area, collaborative applications have not found easy success. The costs of adopting and maintaining new technology to support group work can be quite high, which prevent small or ad-hoc groups from using such systems. A common problem in collaborative applications is a failure by the application developers to understand the extra work imposed on group members in order to use such applications; this failure has led groups to underutilize or even abandon their use of collaborative applications.

The scope of our research focuses on how to support the early stages of the engineering design process. Engineering design is an area that relies primarily on group collaboration to solve most problems - the nature of the work is usually complex and often requires design by groups or teams [50] [38]. The design process not only involves sharing informal information and ideas between group members, but also analyzing opinions and expertise to make reasoned decisions. We are interested in design groups whose members work in different locations and at different times. Our goal was to identify the kinds of support such groups would need to collaborate effectively.

1.1 Collaborative Computing and Related Terminology

Collaborative Computing focuses on the design and development of computing tasks and systems that enhance group interaction and enable groups to cooperate. We believe this term places equal emphasis on the importance of understanding collaborative requirements as well as recognizing the integral role of computing technology in this area. Computer Supported Cooperative Work (CSCW) is another related term that refers to the study of understanding how people work and how technology can support them. It was first coined in the eighties, by Paul Cashman and Irene Greif as part of an effort to improve the realm of computer technology for groups by applying the findings of sociologists, anthropologists, psychologists, and anyone else involved in researching group interactions [20]. CSCW has emerged as a distinct field of research within the last decade. Although CSCW often includes discussion of applications, it is usually considered to be more of a research area. The term 'groupware' emerged to focus on the commercial aspects of CSCW, and is most commonly used in industry [46]. Group Support Systems (GSS) and Group Decision Support Systems (GDSS) are both fields of research that originated from the area Information Systems, and in particular, Decision Support [20] [28] They focus on computing for large groups in the context of meeting support and place a special emphasis on decision making. Most ideas of GDSS have been incorporated into the area of GSS, which is now the common term for both. GSS research has evolved separately from CSCW and has tended to only focus on groups that accomplish work at the same time and in the same place.

1.2 Categorizing Collaborative Applications

Collaborative applications span a broad range of categories, from electronic mail to shared conferencing systems that require customized rooms [53]. The former can be used by groups whose members are in different places at different times, while the latter requires groups to be together in the same place at the same time. Because of this range, it is useful to categorize applications by some set of constraints, both for helping identify

similar applications and for understanding the type of functionality addressed by different classes of systems. A simple and common classification is the Time-Space Matrix [Figure 1], which is described in various forms by several authors [44] [15]. The Time-Space matrix categorizes applications according to how they fall on two dimensions: time and space. The first dimension, time, is defined by the type of interaction that group members have with each other - whether they are communicating in real time or not. Applications that focus on group work that takes place between members in real time are said to be *synchronous*. The most common types of synchronous group-based applications are conferencing or meeting systems. In contrast, applications in which group members do not need to work together at the same time are said to be *asynchronous*. Applications such as those for tracking and viewing electronic discussions are a common example of asynchronous applications. The second dimension, space, is defined by the geographic nature of the users [44]. Group members can be said to be either *remote* or *co-located*. The difference is based on the accessibility of users to each other rather than geographic definitions, although physical proximity often coincides with the classification. The term co-located is used to emphasize this logical division. Electronic mail is a classic example of an application with remote users, whereas systems that require users to gather in one place, such as meeting rooms, exemplify co-located users.

		TIME	
		Synchronous	***Asynchronous***
S P A C E	***Co-Located***	*example*: meeting rooms	*example*: co-authoring systems
	Remote	*example*: real time conferencing	*example:* email

Figure 1-1: Time-Space Matrix

1.3 Further Discussion of the Problems

1.3.1 The Difficulty of Supporting Remote Groups

Collaborative applications require considerable effort to set up and maintain. While most companies have internal networks, use on these is generally limited by firewalls, discouraging interaction from anyone outside of these bounds. This can leave out group members who are consultants, away on business, or working from home. Behavioral studies by Reder and Schwab [39] found that remote access would be required to assure work continuity among such groups. Internal networks can also leave out ad-hoc groups that are formed among employees of several different companies or professors from academia, who may come together as part of a decision making body of a consortium or a conference. In addition to these concerns, ensuring that all group members are running the latest software for their specific platform is a considerable task. Software updates have to be installed on every site where a group member resides, something that can often be handled only by a system administrator. Developing the software requires extensive porting and testing on multiple platforms prior to release. Although networks have made connecting groups viable, there are considerable administrative costs involved.

1.3.2 The Work-Benefit Disparity

Grudin noted that most collaborative applications required people to do additional work to enter or process information required or produced by the system [21] [22]. The extra work is unwelcome because group members do not perceive any individual benefits from it. This observation was borne out in studies by Bullen and Bennett [12], who found that people are not likely to use a tool unless there is a balance between the perceived effort and the benefit to that particular user. An example of this is the use of automatic meeting schedulers: the immediate beneficiary is the manager, but successful use of

automatic meeting scheduling requires additional work for those group members who would not otherwise maintain electronic calendars [22]. Such tools often fail because they require extra work without offering users additional benefits to compensate for it.

1.4 Motivations

Groups often want to work together without being bound by time or place. For example, employees who prefer to telecommute want to work from home yet still need to collaborate with their peers to remain a vital part of their organization. People working on multiple projects and multiple groups need the flexibility of working on each one when they can, without wasting time coordinating schedules among group members. Newsgroups, email and bulletin-board systems have eliminated time and place constraints, connecting people through on-line communities [46]. While these mechanisms support the exchange of information, most of it is largely unstructured. Groups with specific goals, such as design work, need much more support.

Design problems can be characterized as "wicked problems" [43], which have no definitive formulations, no enumerable set of potential solutions, and no ultimate solutions. To address such problems, design teams must coordinate several tasks, which include understanding the problem, generating some solutions for it, and evaluating the solutions to decide on a best course of action. This engineering design process is essentially generalized problem solving [50], and is applicable to various design topics, such as software systems design [38].

1.5 Guide to this Thesis

This thesis presents a framework that describes the types of support that application developers should consider when building applications to aid design groups. In Chapter Two, we discuss other collaborative applications whose work relates to parts of our approach.

Our approach proposes the use of the World Wide Web (web) as a key basis for collaborative systems; Chapter Three presents the advantages of deploying applications through the web.

We have developed a framework that specifies the key supports we believe are necessary to aid groups in web-based collaboration to solve design problems. This framework is presented in Chapter Four. We begin by motivating the need for a framework and how we derived it. We define and discuss the three types of necessary support: Process support, Analysis support, and Interaction support. Together these three supports can help engineering design groups by easing the process for them and by helping them to focus their thinking through evaluative exploration.

Our work began as a result of exploring new approaches to supporting engineering design. In particular, we wanted to extend the design process specified by the EDSS application, a joint venture between the Mechanical Engineering and Computer Science departments here. Because our prototype is an extension of this application, Chapter Five explains the design model implemented by EDSS.

The final stage of our work has been to implement a prototype based on the framework. A description of this prototype is presented in Chapter Six. Interesting implementation issues and observations of our experience developing this prototype are discussed in Chapter Seven. This leads us to the consideration of interesting new work that can continue this research. Chapter Eight presents future work in both the framework as well as enhancing the prototype towards creating a robust application. Chapter Nine summarizes the work presented in this thesis. A bibliography follows, as does a set of appendices, which contain web pages particular to our WEDSS work as well as the reference to the Perl and Java code that implements our prototype.

2. Related Work

Our framework is based on research of two different categories of collaborative applications: ones that focus on information sharing and ones that focus primarily on group decision support.

2.1 Information Sharing

The primary purpose of systems that support information sharing is to create a repository of information that can be shared among group members. Mailing list servers such as Majordomo [34] and Listserv [31] are extensions of one of the most successful asynchronous tools ever - email. Mailing list servers broadcast information to specified sets of users and support limited archiving capabilities. Active Mail [19] is also an extension of email but specifically supports more structured functionality for computer mediated interaction. It records and coordinates collaborative interactions through the look and feel of ordinary email. All of these systems leverage on the success of existing email technology. Most users are comfortable using email and already have the means of sending and receiving email messages. However, the basic architecture of email-based systems limits the development of graphical interfaces and flexible access to shared information.

Several applications support information sharing using databases and have more sophisticated interfaces. The Virtual Notebook System (VNS) is a distributed collaborative hypertext system that organizes shared informal information for viewing by a group of networked users [17]. It uses a simple notebook metaphor to capture and view informal information, which is stored in a database. The interface follows the notebook metaphor and displays text and image information through simple pages that can be connected through hyperlinks (active portions of a page that can bring up other pages). Lotus Notes [33] is a commercial product that supports sharing information through a central repository. The core of Notes is a document database, which can be accessed

through customized forms and views. An extensive development environment provides mechanisms to support collaborative tasks such as scheduling and automating workflow processes. While Notes provides more features than VNS, it also requires more customization and training to use [37]. Both of these products require specialized network setup and maintenance.

The increasing popularity of the web is drawing considerable interest in developing applications that can easily run within web browsers. The growing ubiquity of web browsers promises such applications the accessibility of email while providing richer support beyond simple text. One of the frequently cited advantages of the web was its transparent support for geographically distributed groups using different computing platforms [2] [42]. Systems such as BrainWeb [9] and BSCW [10] support collaborative group idea generation by capturing and recording casual information into web pages that groups can view. The DADAS project is an implementation of the Design Intent framework [2] to facilitate collaborative communication through the web. It was designed specifically to support third party access to a particular listings database. Although the system is intended for a very particular task, its use of the web highlights interesting new possibilities using this medium. Like most of the other systems that support information sharing, it contains a central repository intended to capture and contain shared group information . The output is presented as a web page, and further communication regarding the page or other information can be captured through built in asynchronous communication mechanisms. While some collaborative systems have some kind of notification or update policy, the DADAS system gives a unique consideration to group members by implementing notification that could be personalized by each member of the team.

2.2 Group Decision Support

Our work is also based on models of group decision support that focus on explicitly structuring information to support the design process, such as [30] and [59] as well as those that explore techniques for evaluating ideas and group process, as in [3],

[11] and [55]. Most GDSSs generally support only synchronous, co-located groups and thus rely on off-line, face-to-face communication to support social interaction between group members.

gIBIS [59] is a landmark hypertext system that has found considerable success in supporting designers. It uses the Issue Based Information System (IBIS) model of deliberation [29] to provide a structure that allows designers to easily design ideas and discussion. While it has proven quite successful at representing information, it offers no decision support beyond the stage of structuring information. rIBIS [40] is similarly designed but supports real-time distributed collaboration.

The SIBYL system [30] supports group decision making as gIBIS does: by supporting a specific structure to represent design information. In fact, the author of the system considers SIBYL as an extension of gIBIS. SIBYL differs in that it additionally requires an explicit representation of goals. Enhancements to SIBYL are guided by a design principle derived by the author during the development of SIBYL—that a system which needs to elicit knowledge from people should provide as much support as possible while remaining simple and natural for people to use.

REMAP/MM [3] is yet another system that uses the IBIS model to structure the decision process. Unlike gIBIS or rIBIS, REMAP/MM supports automated reasoning because its implementors viewed this as a critical component for decision support. This system also addresses the interest in multimedia and explicitly supports storing multimedia objects and information about them. While REMAP/MM introduces new notions of multimedia support and reinforces the role of reasoning in decision support systems, it is still restricted to co-located groups.

Several group decision support techniques have been suggested for GDSSs based on research and observation of group decision making processes. Both Bui and Wagner describe the nature of the tasks that groups undertake and present prototypes that solve some of the tasks, such as consensus gathering [11] and process structuring [55]. Co-oP [11] is a research prototype developed by Bui to implement a set of components that coordinate formal group processes. Wagner developed a set of prototype tools, each one addressing a different group task concern.

There are very few web-based group support systems, although several companies have announced plans to introduce such applications [26] [32]. C.A. Facilitator has recently introduced a web adaptation of its PC product called Facilitate.com [16]. This application supports the structuring of decision problems only through chronological entry. The only analytic technique supported is a basic yes-no voting mechanism that simply tallies total votes. While its presence on the web will make Facilitate.com easily accessible to remote groups, its decision support features do not offer enough help to design teams. TCBWorks [48] is a research project developed by University of Georgia which comes closest to offering the kind of decision support required by design teams. It supports a simple hierarchical structuring of ideas and a voting mechanism to support decisions by groups. Unlike C.A. Facilitator, this voting mechanism uses criteria, which can offer insight into the final voting results. Again however, there are no evaluation techniques to explore the results. Although TCBWorks is designed as a system that can be used any time and any place, it has no specific support that addresses either synchronous or asynchronous group interaction, such as a chat facility or notification mechanism.

3. Leveraging the Functionality of the World Wide Web

We believe web-based applications offer great potential for supporting distributed group collaboration. We use the term "web-based application" to refer to applications that are accessed through web browsers and store and retrieve data on web servers using web protocols. Consequently in this thesis, when we refer to the web we imply a combination of the information on web servers, the tools such as browsers to view the information, and the protocols that are used to manipulate the information. The web's origin as a network-accessible information repository [54] is well suited to one of the requirements of group work—connecting users to shared information. The advancements that have been introduced in the last couple of years have made many new things possible on the web. Although web technology is changing rapidly, the current state of the web has the functionality to encourage exploration of group-based collaboration on the web. The web offers an environment that can support: (1) user interaction through graphical interfaces, (2) access to shared data and shared systems, and (3) group work on multiple platforms.

3.1 Rich Interface

Although the internet has long provided access to information on servers connected through it, this was not easy for many users to use. Most users relied on programs such as ftp to explicitly request data transfers to their local machines. Moreover, there was no means of eliciting information from users and responding to them with context specific information. Thus any work that required user input had to be developed on more limited networks that supported such mechanisms, such as X-Windows. The advent of the World Wide Web created a medium that incorporates visual and aural information onto "pages" that can be viewed without users needing to know how to retrieve them. Locations of other pages or data can be embedded within these pages through hypertext [8], allowing users to simply click on links to view or retrieve the related information. One of the most powerful features of the web is its capability for

eliciting user input and acting on it. The FORM elements [35] specifications introduced a set of standard widgets such as textboxes and radio buttons for use on web pages. The Common Gateway Interface (CGI) supports the execution of programs to immediately process such user data and to generate contextual, dynamic information. All of these features combine to make access to the web highly graphical and easily usable through a point and click interface.

3.2 Shared Data and Applications

The architecture of the web makes it a natural environment for group activities, stemming from its origins as a project to enable shared information [5]. Whether the information is located in one physical site or not, the web makes it simple to construct a virtual site that can contain relevant hypertext to all related information. This site is available to anyone with access to the web. Any information posted to a web site is immediately available to others, to be accessed at their convenience. This can eliminate the need to distribute data any further than the local web server; interested users can access the information they need, when they need it. Applications deployed within web pages benefit even more from this shared model. Like data, such applications can be used by anyone on the web without requiring them to download and run a version of the application on their local machine, or connect to specific machines to use the application.

3.3 Multi-platform Support

Web browsers handle platform dependent issues and support common protocols that render data independently of platforms. Anything that can be viewed within a web browser on one platform can be accessed just as well through another browser on a different platform. Although at this time not all browsers support all features, there is ongoing work in the web community to develop a common set of standards that all browsers will minimally support. Consequently, web-based applications can be implicitly

supported on multiple platforms without the usual development and maintenance costs that standalone applications incur in trying to run on different platforms. This advantage makes it easier for developers to build applications for heterogeneous groups since they only need to develop a single version of the application to be used on a variety of platforms. Groups especially benefit because they are not forced to use a particular platform to use an application. Distributed groups are even more likely to work on a variety of platforms so multi-platform support is even more critical to support them. Supporting functionality on multiple platforms allows group members to collaborate on common problems while still leaving them a choice of working in the environment in which they are most comfortable.

3.4 Extensions to the Web

Continued exploration of web technology has introduced new features and ideas that can enhance the functionality of applications deployed on the web. Several researchers have experimented with extensions to browsers to support real time synchronization [18] [58]. Another project has focused on extending FORM elements to provide client-side form management [49]. Commercial enhancements continue to change the look of the web. New versions of browsers support more and more types of data, such as video and animation. Additions to HTML [25] such as TABLE and FRAME tags can enhance the look of pages. The popularity and potential of the web promises continued interest by both the research and industrial communities.

4. The Framework

We have developed a conceptual framework to describe the kinds of support that are critical for helping design teams collaborate through the web. The purpose of the framework is to make application developers aware of what they should consider when designing collaborative applications for engineering design groups. In particular this framework focuses on web-based groups, which we define as groups whose members are distributed anywhere in the world and who work asynchronously, using the web as a common access medium. The three key supports specified by the framework are process support, analysis support, and interaction support, each of which will be discussed in detail later in the chapter.

The development of this framework was motivated by the need to identify what should be done to support distributed engineering design on the web. The literature in the fields of CSCW and decision support yielded few systems that supported the informal nature of the early stages of design; [3] and [30] were among the few exceptions. Even fewer supported distributed work (except [40] and [59]) and despite the frenzy of development of groupware on the web [16] [26] [32], no systems capture and structure design information *and* support any analysis beyond simple voting schemes. As new technology makes more sophisticated computation and interfaces possible on the web, it seems reasonable that such systems will be developed. However, developing new applications will not automatically provide good computer support for groups [6]. If application developers are to effectively support group design, they must be aware of what things must be supported.

We reviewed literature from the areas of CSCW and decision support to explore research findings and examine experience reports to identify characteristic elements that could be essential to supporting engineering design. Some work related to group process theory yielded new considerations of group interaction that seemed to be overlooked outside of its particular field. We used the information gathered from these areas to help us synthesize various concepts, observations, and ideas into one framework that describes

the functionality that should be included in applications that support design work. In the rest of this chapter we discuss each particular type of support specified in this framework.

4.1 Process Support

Webster's Dictionary defines process as "a series of actions or operations conducing to an end" [56]. The actions and operations of engineering design teams center around sharing ideas to solve design problems and evaluating those ideas. Each group member must not only contribute his/her own ideas but also respond to others' solutions. This deliberation of the problem is as much a part of the solution as the solution itself [24] [50]. The notion of process then is a set of goal-oriented tasks as well as a representation of the design decisions that accomplish a group's goals. This representation should be captured and maintained with other group and design information. Application developers can actively reduce work-benefit disparity by simplifying some of the process tasks group members engage in to work together. Collaborative applications are often more difficult for users to understand how to use correctly; web-based groups are especially vulnerable since there are no common use models for collaborative work on the web. Application developers should consider how to make process tasks more visible and understandable to such groups.

A group's design decisions include both the information or ideas generated by each member as well as discussion of these ideas. Group memory refers to this body of knowledge as well as any mechanisms for capturing and recalling it [24]. This concept of group memory (also referred to as organizational memory when applied on a larger scale) originated primarily from Group Support Systems research, but researchers interested in decision support are slowly recognizing the value of preserving the rationale behind decisions [51] [59]. Asynchronous distributed groups must rely only on the information stored in group memory to work together since it is their only source of shared information. Unlike synchronous groups they can not immediately obtain information that is missing from the discussion, and unlike co-located groups they can not turn to the

group member sitting next to them for help. The application developer must consider how and what to capture to support group memory for web-based groups.

Collaborative work requires group members to stay involved in the process and contribute when they're needed. Traditionally, Group Decision Support Systems have relied on same time and/or same place systems, where the state of the work is known at all times, by virtue of members participating at one time and being in one place where they can easily survey progress. Distributed groups need a mechanism for maintaining this awareness across boundaries of distance, and asynchronous groups must even have a way to stay aware across boundaries of time. Web groups are challenged by trying to stay informed about the current state of a design discussion, since group members can update the state at any time. Without explicit support for this, systems place an extra burden on group members by forcing them to do additional work to stay aware of progress in their group's design process. Groupware has failed in the past because application developers did not account for the extra work needed by each member to support the process, work that individuals were not willing to undertake.

Groups operating towards a common goal depend on each group member to understand how to perform the necessary actions and act on these to achieve the group goal. Collaborative applications are often difficult for group members to use, since they involve a different model of use than single-user applications [37]. Many researchers fail to remember or consider necessary training in their initial prototypes, yet note the need after group studies [4] [37] [59]. Web groups are particularly vulnerable to this since they can not turn to other local peers for guidance. Thus application developers should consider the complexity of their use model and provide support that will help users understand the process.

4.2 Analysis Support

Analysis in the context of the design process is the evaluation of information to find and understand potential solutions. Engineering design groups engage in analytic tasks to aid them in understanding design problems and making good design decisions.

Computer support of these tasks can benefit design groups in several ways. Tools that automate decision support enable groups to utilize decision theoretic methods without requiring members to be experts in formal decision processes. The presence of such tools can also make an application more appealing to groups. Web-based groups especially benefit from automated tools since these can be made easily available to all group members. We focus on two types of tasks that support analysis: (1) decision analytic methods that yield comparative results, and (2) exploratory methods that engage group members in problem understanding. Although the theoretical basis for these tasks can be found in literature, their value to web groups depends on application developers who can effectively translate their functionality through a well-designed interface that simplifies input entry and makes output results easily understandable.

Web-based groups can especially benefit from analysis support because such support can provide decision analysis techniques that can be accessed by all group members. A well-designed interface ensures that everyone in the group can take advantage of decision analysis without needing to be familiar with complicated computational methods. Common access to the same computational mechanisms within the application standardizes the use of analytic methods and reduces the possibility of individual computation errors.

Decision Support Systems have been developed since the early seventies, for a variety of different problem domains, and there are many decision models and methods available. For example, Ullman and D'Ambrosio found seven different types of tools they could classify as decision support tools for engineering design alone[52]. Decision analysis helps groups to rapidly and conveniently analyze options. Although application developers can find many techniques to aid the decision making tasks of engineering design groups, they should consider which methods are most suited to web-based groups. Since such groups are expected to work asynchronously, it should be possible for any group member perform decision analysis without depending on participation from others. Decision analytic techniques will benefit groups most if they can perform elegantly even with incomplete information. The results of such techniques should be easily displayed within

web browsers. The incorporation of decision analysis into web-based applications makes powerful theoretic techniques available for practical use.

Although decision making techniques help groups compare possible solutions, they give no insight into the rationales behind the solutions or the factors that might have influenced them. Constructing an understanding of the solution space is as important for web-based design groups as defining the solution space. These groups rely on each member to participate intelligently in the design discussion. To do this, group members must often construct their own model of the relationships between factors that affect the design problem. While more experienced members may do this easily, others could have difficulties. Supporting tasks to help users make sense of problems gives groups opportunities to use common methods to develop an individualized understanding of the problems. This construction of meaning is similar to the concept of sensemaking [57] found in group support systems research. Sensemaking tasks are a set of processes that help reduce confusion arising from a misunderstanding of the problem. Sensemaking tasks are intended to help groups frame the problem and derive its causal structure. While Group Support Systems research uses sensemaking aids to help co-located groups, the goals of sensemaking seem well matched to address web-based groups. Supporting this seems a natural task for collaborative engineering design systems to include, where part of the final design solution is a better understanding of the design problem.

The application developer's job is to choose appropriate techniques for the types of analytic tasks described earlier and translate them into usable approaches. The appropriateness of a technique should include consideration for the kind of user effort that will be required to utilize a method or process the results of it. A well-designed interface can put theoretical models into practical use [7]. Analysis support should adapt decision theoretic techniques as well as aids for sensemaking types of tasks to web-based groups so they can benefit from analytic methods usually reserved for synchronous, co-located groups.

4.3 Interaction Support

Good engineering design depends on groups who can share ideas, perspectives, and expertise. This interaction between people can be the source of many benefits of collaborative work. Although research in group process theory has identified many gains from group work [36] such as stimulation (group members feed off of each other) and learning (members imitate and learn from more experienced ones), few tools have been explicitly designed to encourage interaction to achieve these specific group gains. Many collaborative applications assume interaction occurs off-line apart from the application, or they supply synchronous support mechanisms such as chat tools. The first ignores the role of interaction in the overall process and the second provides a means for communication but does not actively help interaction. Good interaction can help groups to learn about each other as well as from each other. Web-based groups do not have the infrastructure for interaction that is implicitly available to face-to-face groups. The results of interaction between group members is what makes some group gains possible. Thus it is more difficult if not impossible for web-based groups to achieve the same group gains enjoyed by traditional groups. Interaction should be supported by explicit means that make group gains achievable for web-based groups.

Traditional groups become familiar with each other through informal encounters at work and business situations such as meetings. These experiences help them to gain a sense of the personalities of members in their group and gives the group a common background. This familiarity can help group members work together by supporting a feeling of community. Because web-based groups do not have these opportunities for acquaintance, interaction support can seek ways to help groups share this type of information. Application developers should consider both how to obtain and how distribute this information to groups. As with other support tasks, ease of use is critical, to avoid burdening group members with extra work. Explicit support that allows group members to become acquainted with each other can introduce a sense of community that can foster good group relations. An example of this is simply storing some personal information about each group member which can be quickly accessed by other group

members to give them a chance to become familiar with each other when physical encounters are not possible.

Both individuals and groups benefit when group members can learn from each other. The value of leveraging the knowledge of experts seems to be becoming a topic of interest. Applications such as Answer Garden [1] promote learning gains by building a group memory based on expert information and advice that users can access for consultation, and contact experts when needed. This functionality is well-suited for web-based groups whose members may not be familiar with each other's areas of expertise to know who or where to turn to. There are two components that application developers should consider supporting to motivate learning gains: (1) making expert opinions easily available for users to learn from, and (2) helping group members identify experts within the group that they can turn to for personalized help if necessary. The first encourages passive interaction by letting groups learn by observation and the second provides active interaction by directing group members to initiate contact with other group members through a common context.

4.4 Summary

This framework describes the three types of support that we believe should be included in collaborative applications for web-based design. Process support, analysis support, and interaction support address the requirements of the group design process in a web-based context. Ideas described in each type of support were derived by an examination of the literature in several research areas that share a common interest in collaborative work. Process support centers on defining a good group memory system and identifying ways to simplify auxiliary process tasks. Analysis support handles decision outcome calculations as well as sensemaking tasks. Explicit support for interaction is critical in helping web-based groups achieve some of the same group gains that traditional groups implicitly enjoy. The framework integrates many different types of support because web-based systems require much more explicit, automated task support than most collaborative systems.

5. The EDSS Design Model

We used the framework to modify an existing research system (EDSS) that was developed to support the engineering design process in a meeting room situation. Before describing our prototype, some background information about the original system will make the features it supports more understandable. In this chapter we describe the design model for EDSS, a joint collaboration project between the department of Computer Science and Mechanical Engineering at Oregon State University.

EDSS (Engineering Decision Support System) is a group collaboration tool targeted to support design engineers in the decision making process. Design problems are captured and organized using the model of Issue Based Information Systems (IBIS) [29]. Analysis of the problem representation is supported by incorporating decision theoretic semantics into EDSS to accommodate evaluation of solutions. These evaluations utilize mathematical methods that allow EDSS to produce usable analysis without requiring a mapping of the complete solution space. This is of particular benefit to engineering problems, where many design decisions must often be made early in the design process where complete quantified information is not available [50]. In accordance with the classification of collaborative systems as described in Chapter 1, EDSS is a same-time, same-place system because the information can only be stored and accessed from one machine and is intended for use as a support tool in real-time group collaboration meetings.

5.1 The Elements of EDSS

5.1.1 Issue Based Information Systems (IBIS)

As stated earlier, EDSS follows the IBIS model to capture and organize its representation of the design space. This model was originally proposed by Rittel in 1970

and structures information by defining a simple categorization for discussion of decisions. IBIS has been used in several projects such as gIBIS [59], rIBIS [40], and REMAP/MM [3]. It has been found to be an effective way of representing information with a minimal amount of formal structure. IBIS separates information into three basic types: Issues, Alternatives and Arguments.

5.1.1.1 Issues

Issues state questions or problems that are to be solved. In the context of EDSS we consider issues to be design issues, which are any problems identified by the design group that need to be resolved. While issues can be interrelated, the focus of current work in EDSS as well as our work in this thesis is on single issue problems.

5.1.1.2 Alternatives

Alternatives (also referred to as Positions) state the possible resolutions of an issue. Again, because the goal of EDSS is to support the design process, these alternatives are particular options generated by members of a design team to solve a particular design issue.

5.1.1.3 Arguments

Arguments are rationales for either supporting or opposing a particular alternative. Arguments are evaluative and made by comparing alternatives against requirements (EDSS formalizes these requirements as Criteria, which we discuss in the next section). These evaluations can be made by any members of the design team, and are subject to their personal levels of expertise and opinion.

5.1.1.4 Limitations of IBIS

Experiences with IBIS have shown that even the simplest attempts with such a representation of problems have yielded considerable benefit in supporting decision making [52]. Yakemovic and Conklin found that the issue based approach helped users to frame problems and invent new solutions [59]. While this approach has been successful in

creating an effective arena for information representation, IBIS does not specify any support for choosing among alternatives. In surveying many decision support systems [52], Ullman and D'Ambrosio suggest that systems that support only representation should be encouraged to offer more decision support.

5.1.2 Decision Theoretic Semantics

A primary focus of EDSS was to develop decision theoretic semantics that would allow automated tools to provide stronger support for design teams. While the IBIS model provides a solid structure to capture representation, its semantics are too weak to support evaluation [23]. The decision model for EDSS adds three components to overcome these weaknesses: Criteria, Confidence and Knowledge.

5.1.2.1 Criteria

Criteria are the requirements, specifications, or constraints that measure the Alternatives, or proposed solutions. Criteria contain context-specific requirements derived by group members. These are used to evaluate the feasibility of the current set of Alternatives suggested for a Design Issue. The impact of the Criteria on the Alternatives can be personalized by each group member by weighting each Criterion.

5.1.2.2 Confidence

Confidence quantifies the level of surety about a propositional assertion made than an alternative will satisfy a criterion [23]. It represent an individual's personal belief of the potential success of an alternative in satisfying the criterion. This is a subjective evaluation based on an individual's own understanding and analysis of the problem.

5.1.2.3 Knowledge

Knowledge captures the level of expertise that each individual brings to the discussion. It represents an individual's personal expertise about a specific alternative with

respect to a specific criterion [23]. An individual's knowledge level about an alternative is independent of how well he/she thinks it will satisfy the criterion.

5.2 Calculations in EDSS

The representation of the design discussion captured by the Knowledge and Confidence components is converted into numerical values that quantify this information. Expected Utility values are calculated from these to indicate individual and group preferences for Alternatives. Group members can examine their preferences for various Alternatives by comparing the Expected Utility values, which are calculated using Bayesian decision theory [45]. These values represent a user's indication of satisfaction for the alternative in question. Unlike other decision comparison methods, this model can derive results from incomplete or abstract information; that is, generating a utility value for a particular alternative does not require that it be evaluated by every group member or even be considered with every criteria.

For further details of the calculations used in EDSS, please refer to [14] and [23].

5.3 Summary

EDSS not only provides a means of easily representing information but also incorporates support for making and recording decisions directly related to the information. The addition of decision theoretic components to the IBIS model provides a way to specify a structured set of constraints in the form of Criteria and to capture qualitative considerations when evaluating constraints against potential solutions (Alternatives). A model of the components and their interrelations is shown in Figure 5-3. Alternatives and Criteria are evaluated as pairs (i.e., A1-C1), which contain information about a group member's evaluation of that particular pair. This information includes the argument explaining the evaluation, a Knowledge level of the expertise of the group

member, and a Confidence level indicating belief that the Alternative will satisfy the Issue for the indicated Criterion.

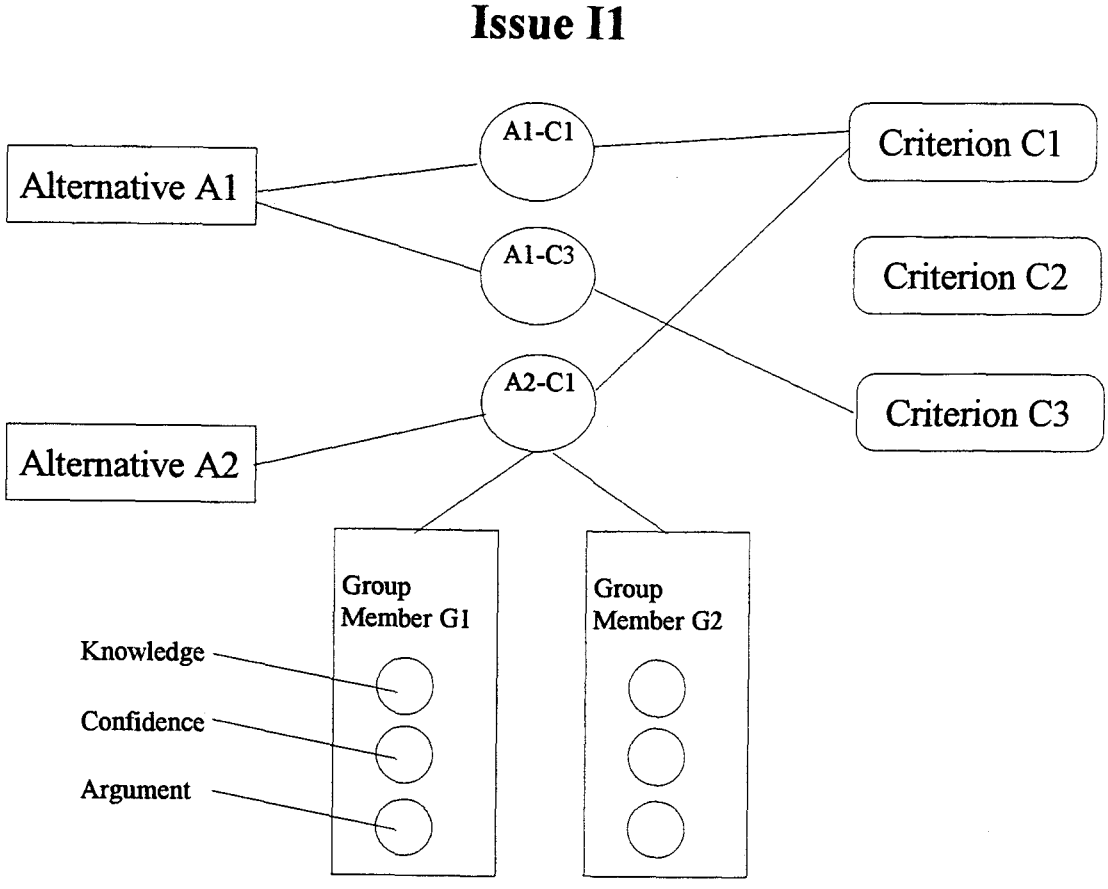


Figure 5-1: Issue Map

6. Implementation of the Prototype

We have developed a prototype of a web-based version of EDSS that incorporates the three types of support defined by the framework discussed in Chapter Four. Our motivation for developing this prototype was to see if we could implement concrete examples of process support, analysis support, and interaction support by using the ideas described in the framework. The prototype, WEDSS was a two stage process—we first developed WEDSS 1.0, which adapted the functionality and look of the PC version of EDSS to the web. We then designed and developed WEDSS 1.1, which added a set of agents to execute concrete tasks that implemented the three types of support. In this section we will first describe the interface and functionality of WEDSS 1.0 which we then use as a basis for discussing the second part of our implementation work—WEDSS 1.1, which is an example of one way our framework can influence implementation.

6.1 WEDSS 1.0

WEDSS 1.0 was developed from observation of the functional requirements of PC-EDSS and discussions with developers of PC-EDSS. While most of the web pages that comprise WEDSS 1.0 map to screens in PC-EDSS, changes were made where necessary to accommodate constraints imposed by the web interface. Before discussing these and other implementation issues, we will describe typical user interactions with WEDSS 1.0 to guide the reader through the application interface.

6.1.1 WEDSS Login and Registration

A user begins a session with WEDSS by accessing the WEDSS Home Page where she/he specifies a login name and selects a group database [Figure 6-1]. Each database simply represents a way of keeping information partitioned between groups, which provides a distinct decision space for each group.

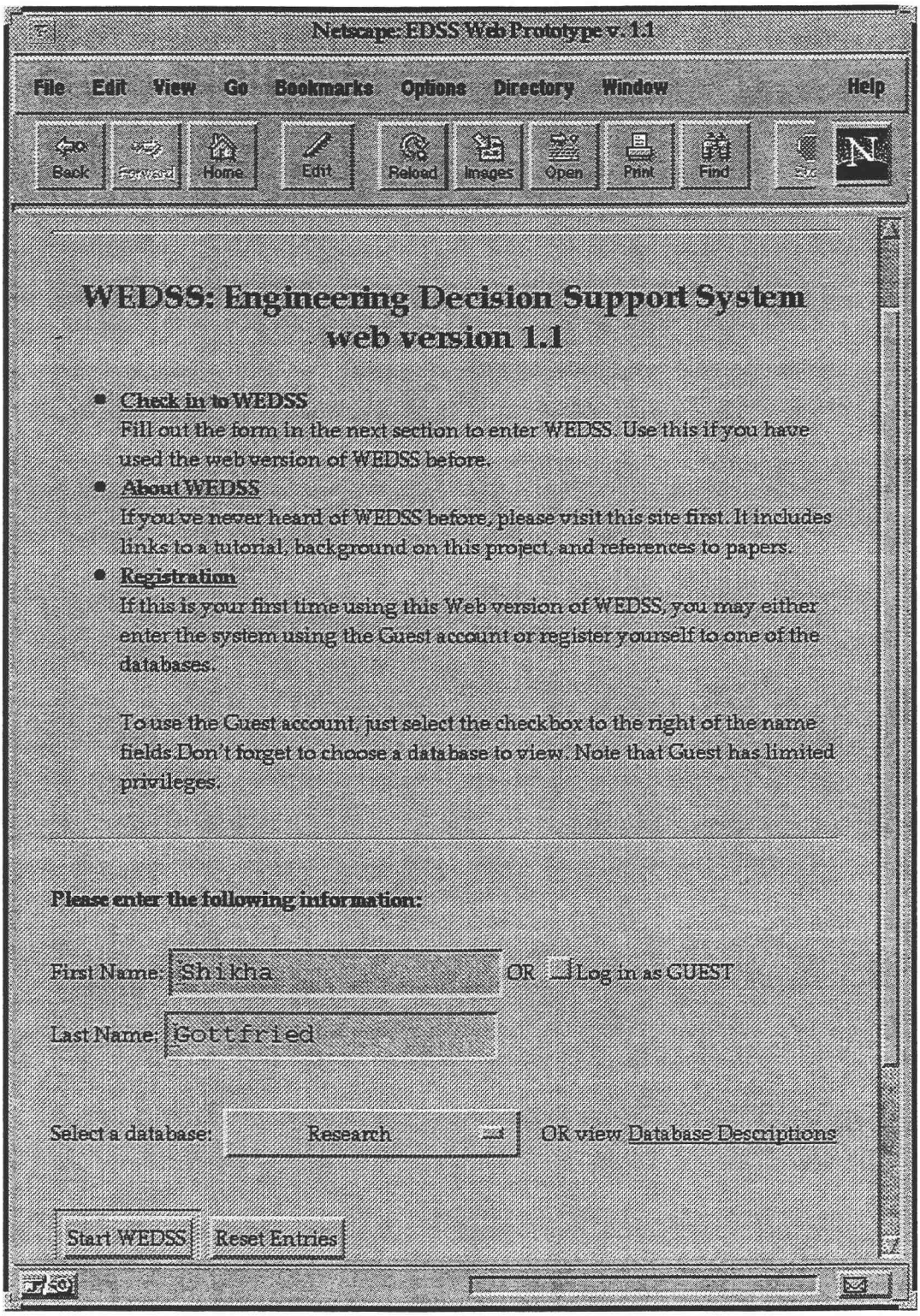


Figure 6-1: WEDSS Application Home Page

While databases can be restricted when groups require privacy, users can currently select any group database, or they may use a Guest account to view any group databases in a read-only mode (participation in the design discussion is prevented). Users can register themselves with WEDSS through its automatic registration form [Figure 6-2].

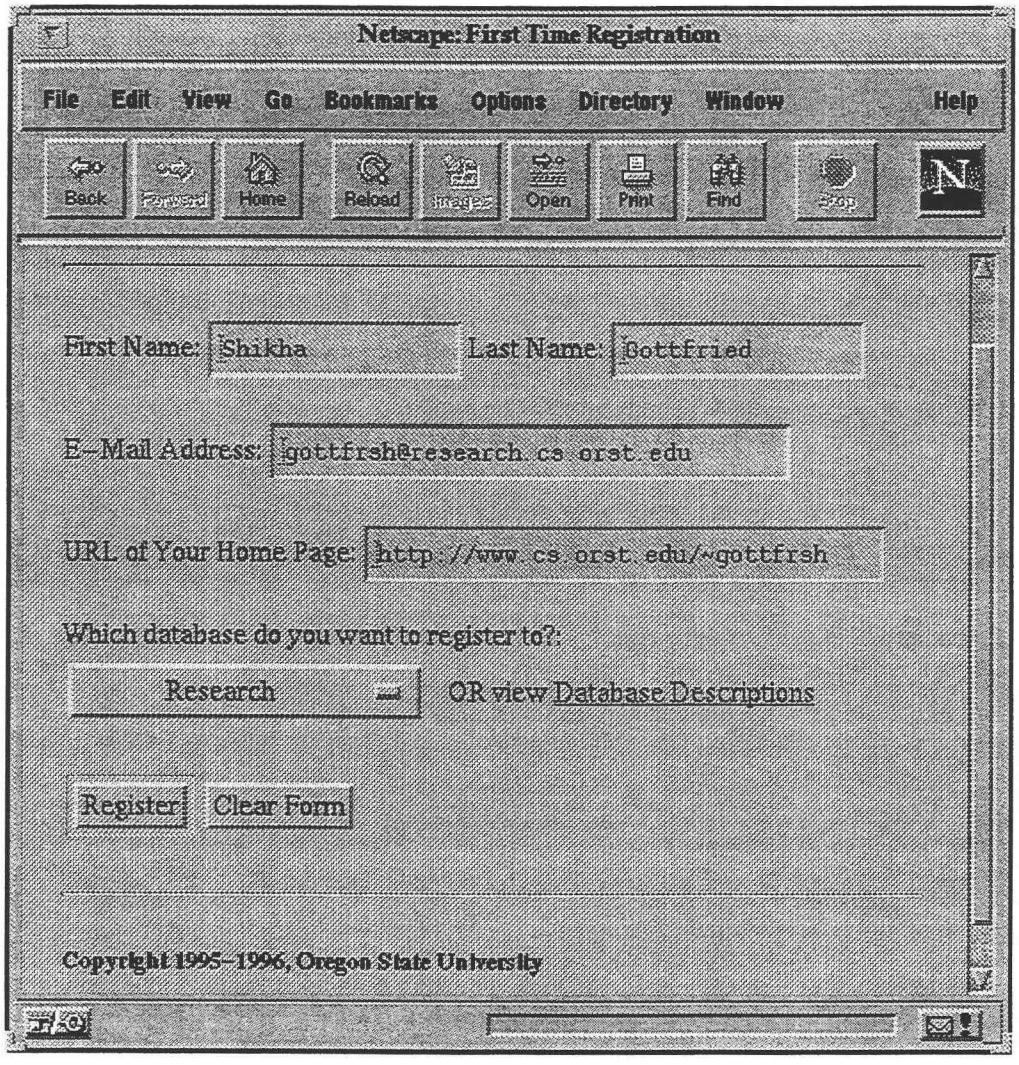


Figure 6-2: Automatic Registration Form

6.1.2 Choosing an Issue

Once a registered user logs in to WEDSS, she/he is taken to the Issue Page [Figure 6-3] which offers the following choices of actions: (1) select from the current list of issues to participate in one [Figure 6-3(a)], (2) select the Modify Issue hyperlink to modify an existing issue; this accesses the list of issues that can be modified (users can only modify issues they have authored) [Figure 6-3(b)], (3) press the "Show Description" button to view the details of an issue such as who authored the issue and a description of the issue [Figure 6-3(c)], or (4) complete the bottom section of the page to add a new issue [Figure 6-3(d)].

6.1.3 Evaluating Alternative-Criteria Pairs

After choosing an issue, the user is taken to the Evaluations Page [Figure 6-4], where most of the evaluation input is entered. Although this page is primarily used by group members to enter their evaluations of alternatives, it also supports access to other features, such as hyperlinks to pages that will add new alternatives or criteria or modify existing ones.

To evaluate an alternative with respect to a criteria (Alternative-Criteria Pair), a user must choose an Alternative and a Criterion via the pull down menus, then mark the appropriate Knowledge and Confidence ratings to indicate how qualified the user is to judge the AC pair and how confident the user is that the particular solution will satisfy the issue. Figure 6-5 shows an example of an evaluation entry. A free form text area captures any informal design rationales the user provides for the associated Alternative-Criteria pair. These rationales, along with Knowledge and Confidence levels, can be viewed by any user at a later date. An advantage of the web interface is the ease with which references to external information can be embedded into the rationale statements, giving other users convenient access to related information on other web sites.

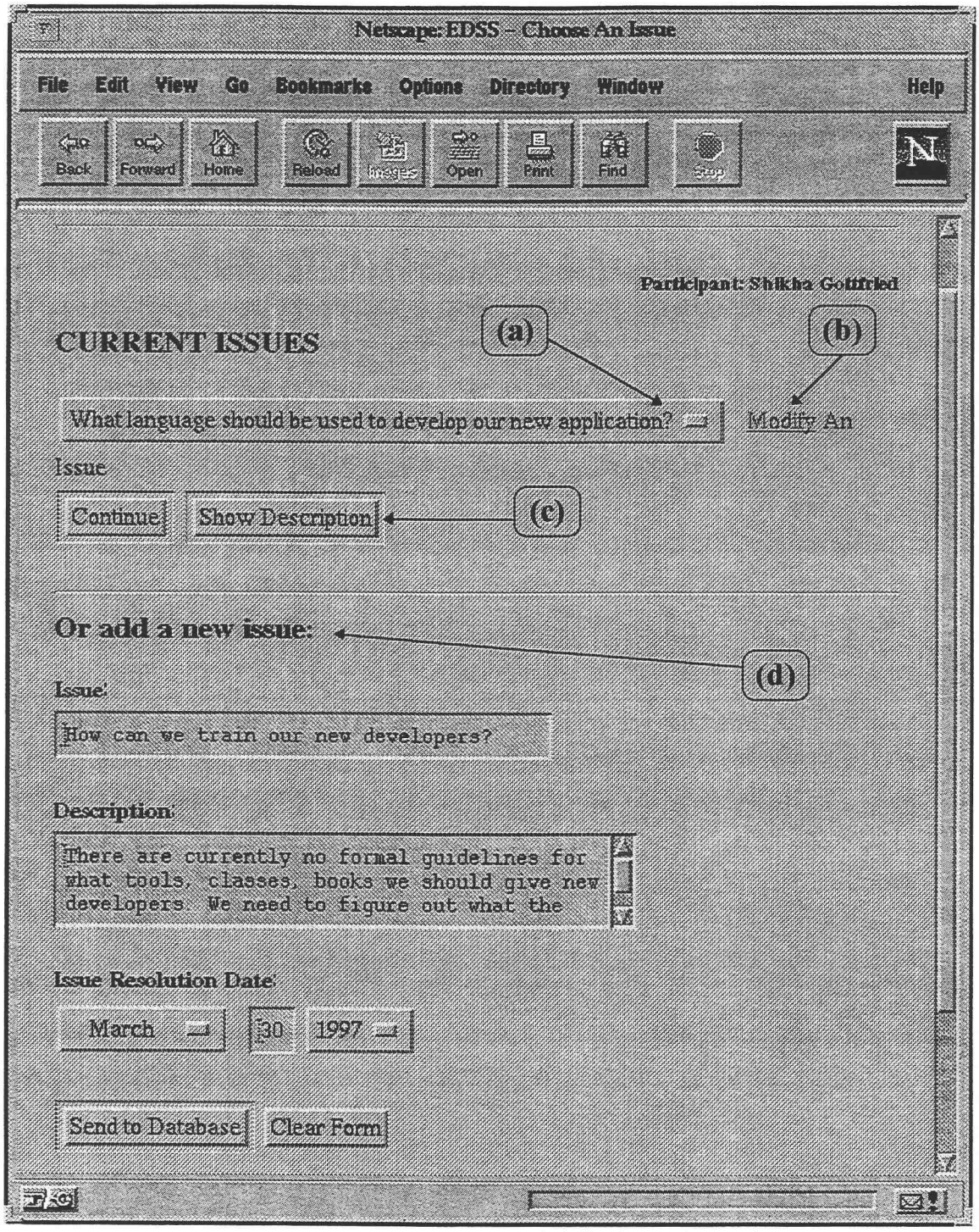


Figure 6-3: Issues Page

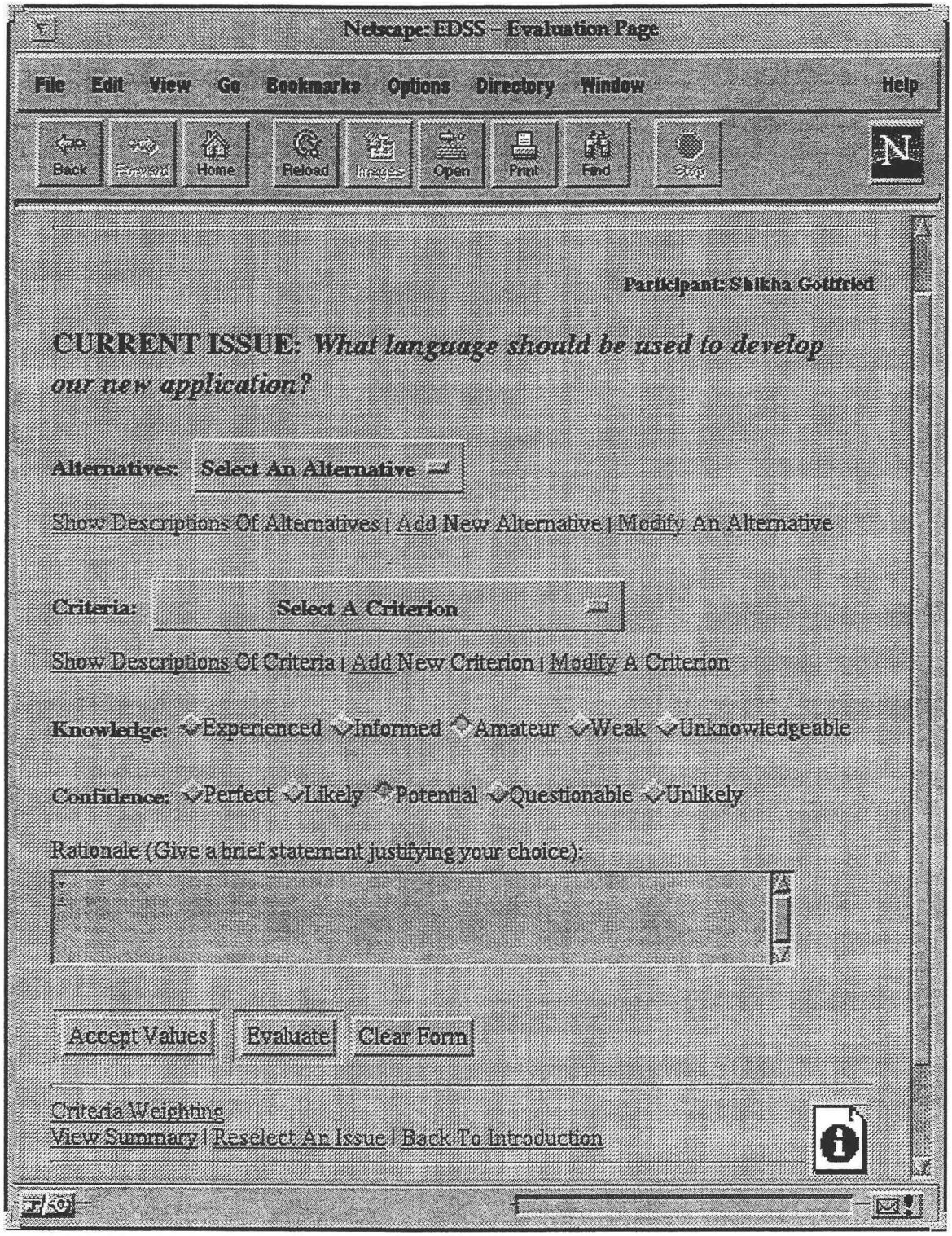


Figure 6-4: Evaluation Page

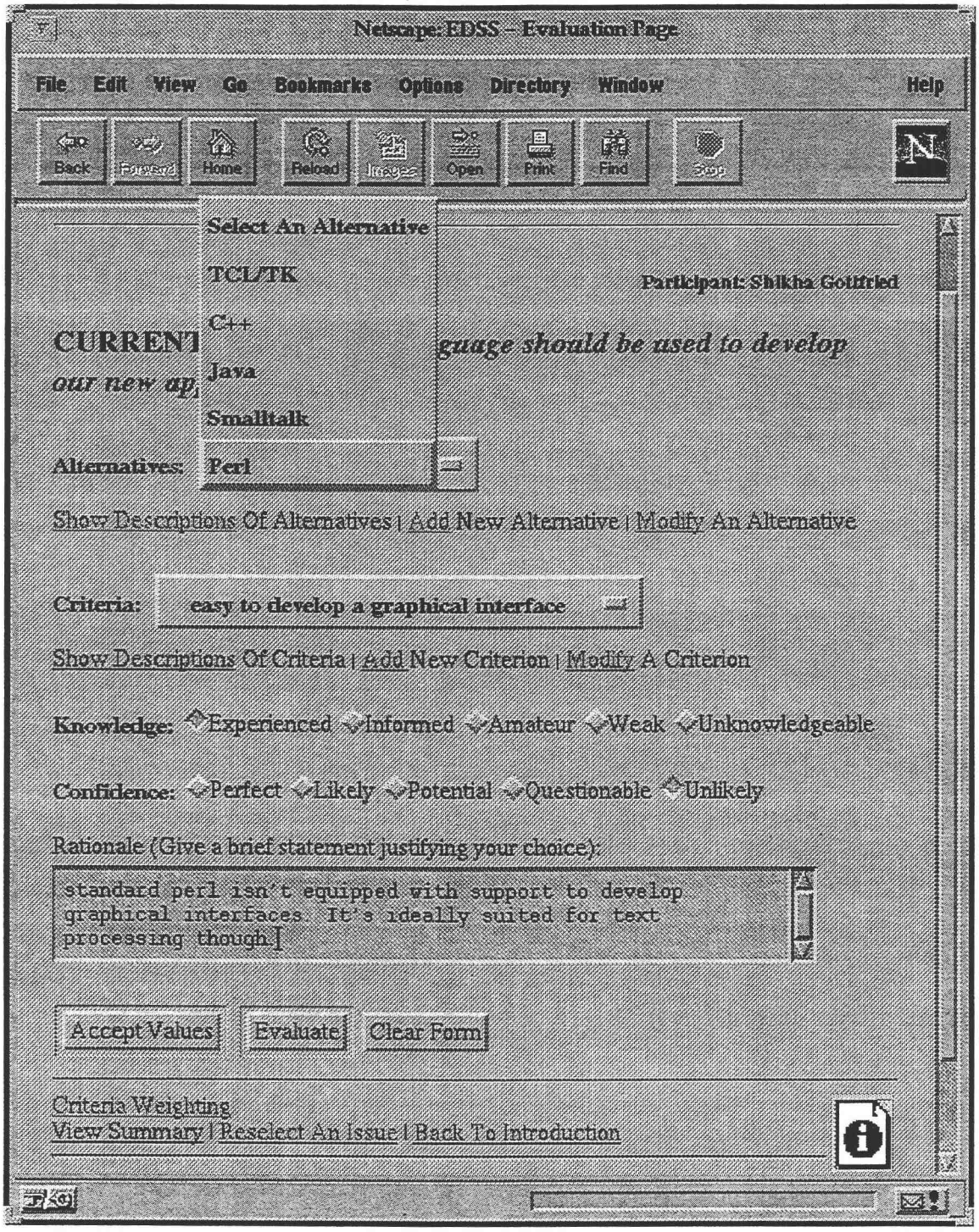


Figure 6-5: Example of Evaluation Entry

6.1.4 Analyzing Issue Results

The current state of an issue can be analyzed at any time by pressing the "Evaluate" button at the bottom of the Evaluations Page. WEDSS then generates a table [Figure 6-6] listing the expected utility values for each evaluation participant along with a group average value. These values show each user's preference for various alternatives, as well as an averaged group preference based on all group members who have participated

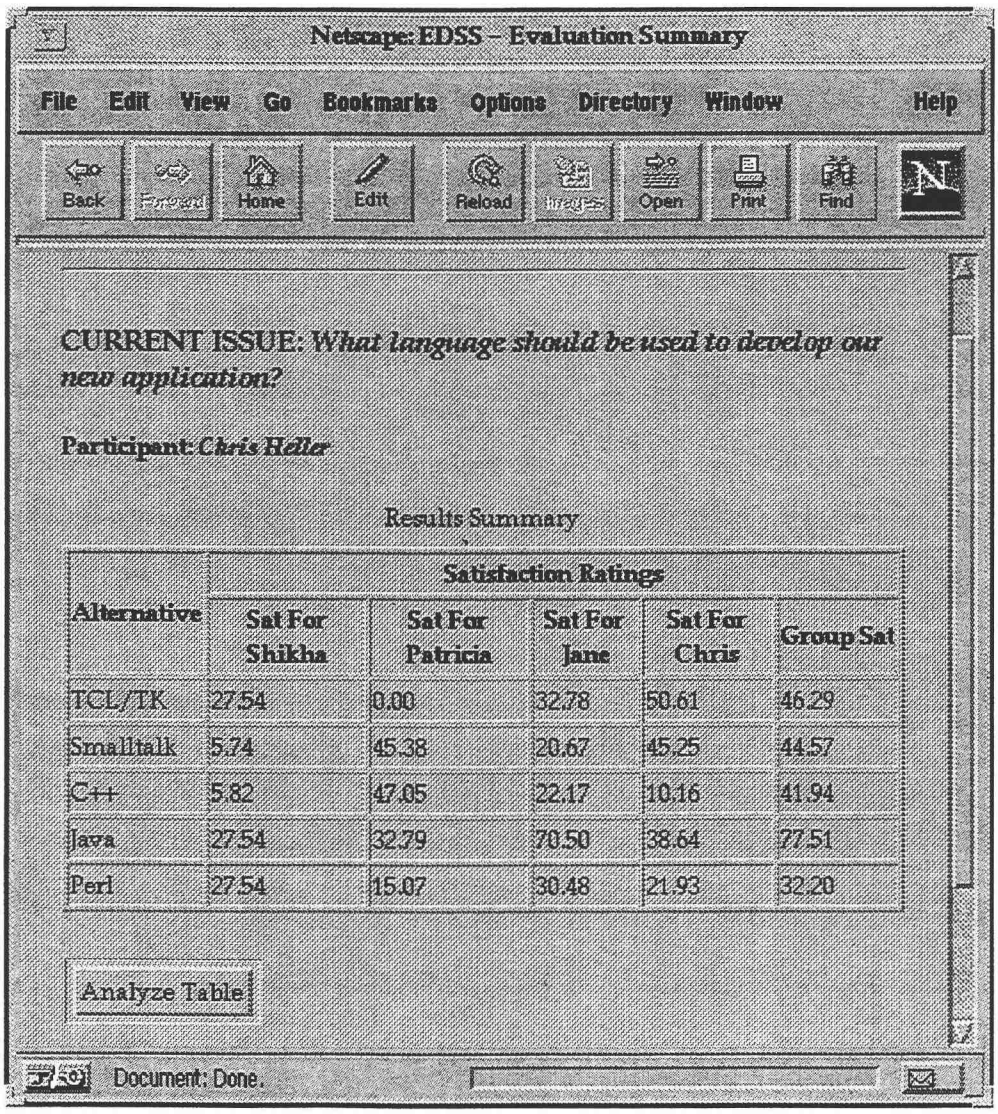


Figure 6-6: Evaluation Summary Page

in any evaluations. Users can keep track of their evaluations using the View Summary Page [Figure 6-7], which shows which Alternative-Criteria pairs have or have not been evaluated. This summary table includes the design rationale as well as a user's Knowledge and Confidence values for every evaluated Alternative-Criteria pair. Users can also view other group members' evaluations via this View Summary page.

6.2 WEDSS 1.1

WEDSS 1.1 extends WEDSS 1.0 by incorporating mechanisms for further analysis support and adds functionality for process and interaction support. We developed a set of agents to execute specific tasks that would provide these types of support. Because the framework imposes no specific implementations of support, developers are free to make those choices themselves. Our experiences developing WEDSS made it apparent that there were many possibilities for useful tasks to automate beyond ones that we chose. It is quite possible and probably likely that several different agents would be needed to truly provide any one type of support. Thus, in addition to developing specific agents, we introduced the idea of an Agent Manager in WEDSS 1.1, which manages the common interface interactions and simplifies the process of adding and removing agents.

6.2.1 The Agent Manager

The Agent Manager manages all agents that display results within WEDSS. It provides the view [Figure 6-8] that users see when they press the "Analyze Table" button after looking at the Expected Utility values [refer to Figure 6-6]. The Agent Manager handles general interactions between a user and these agents, which helps to provide a common and consistent interface. By controlling code common to the agents, the Agent Manager allows agent developers to focus on their specific agent, which can then be easily added into the structure of the Agent Manager. This common code encompasses three significant parts of agent management: (1) the common interface, (2) agent initialization,

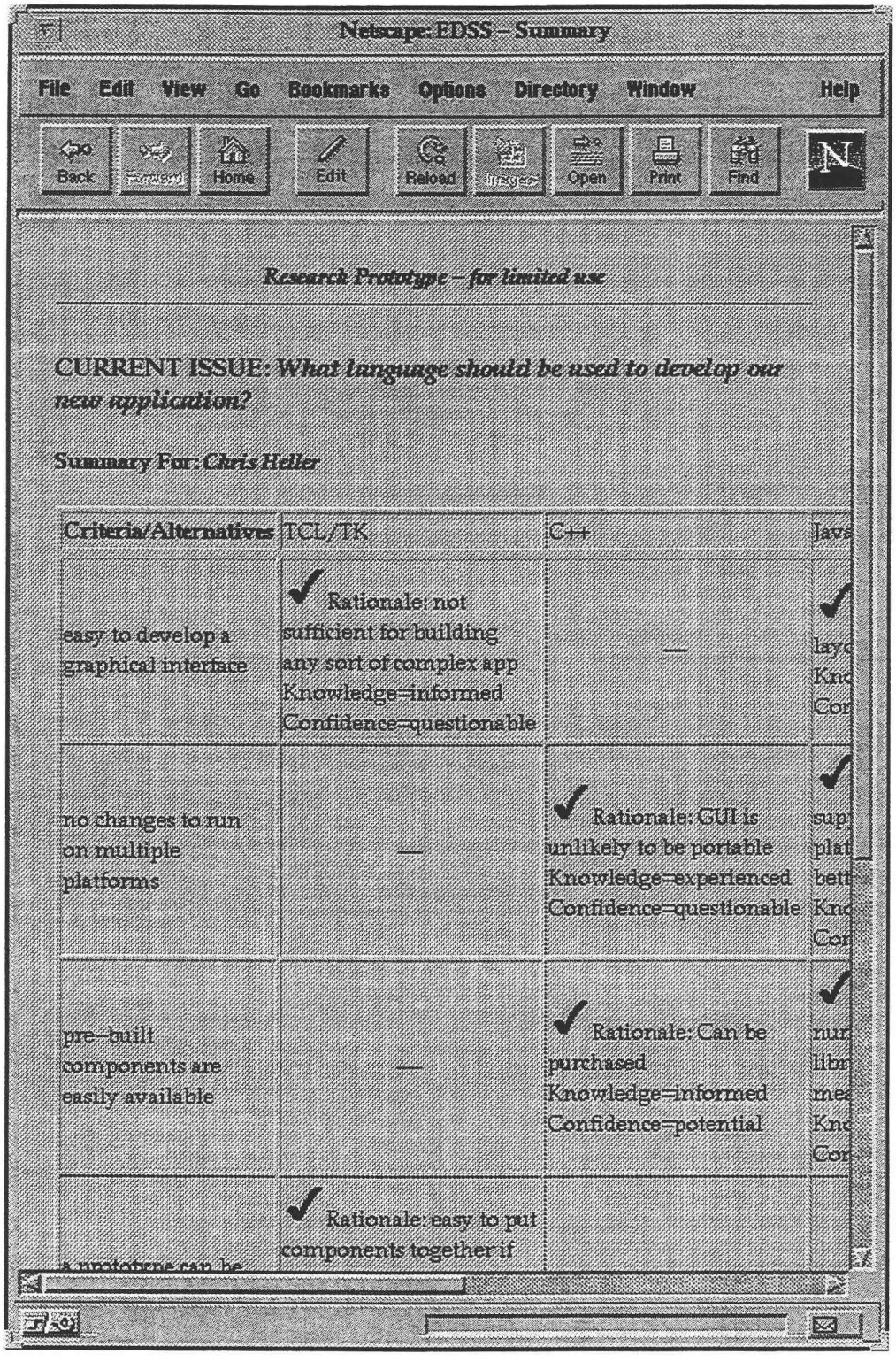


Figure 6-7: View Summary Page

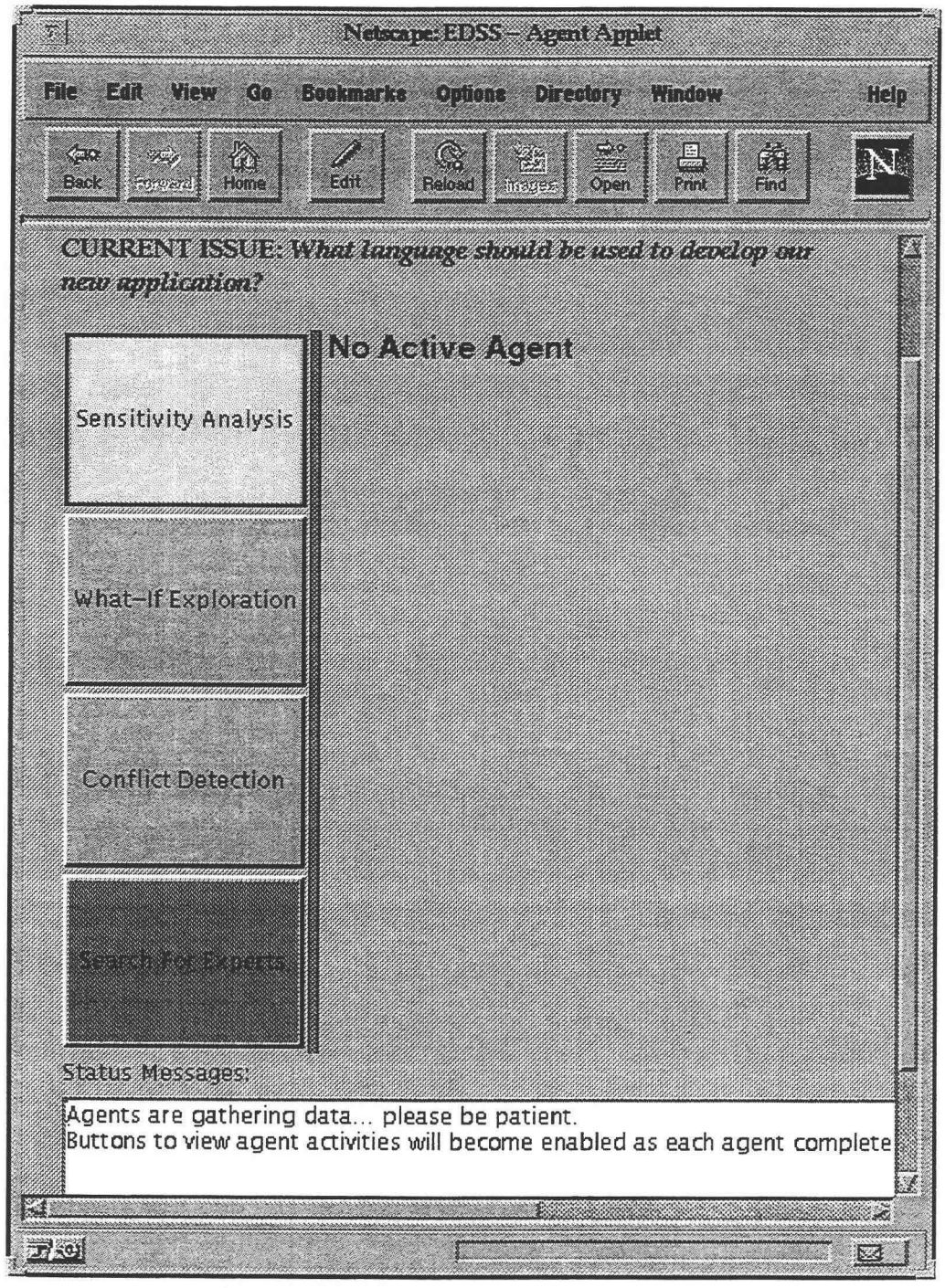


Figure 6-8: The Agent Manager

and (3) the connection to the WEDSS databases. Adding or removing agents from WEDSS can be accomplished simply by making a few small modification in the Agent Manager. Agents are launched by the Agent Manager but run simultaneously yet independently of each other. The Agent Manager initiates and controls contact with any databases in WEDSS. It provides a layer between the agents and their access to the data, which prevents agents from being dependent on the type of database used in WEDSS. Supporting different databases such as Oracle or SQLPerl (used in the current implementation) do not require rewriting code in each agent, only changes to the common database access code in the Agent Manager. The Agent Manager provides a structure for WEDSS agent developers to easily integrate agents into WEDSS.

6.2.2 The Agents

6.2.2.1 Process Support: Notification Agent

The Notification Agent's tasks are to: (1) monitor the databases in WEDSS for changes to the discussions, and (2) report changes of interest to group members. User preferences for which changes in the discussion might interest them are first captured [Figure 6-9] when a user registers into a WEDSS database group. Modifications can be made to these preferences at a later date through General Options [Figure 6-10]. The agent checks for additions or modifications to any issues, alternatives, or criteria. It accomplishes this by periodically scanning the databases in WEDSS and generating customized updates for each user based on his/her interest in the types of changes to be notified for. These updates are then sent via electronic mail to each user. In WEDSS 1.1, the Notification Agent scans the databases nightly for updates.

6.2.2.2 Analysis Support: Exploration Agent

The Exploration Agent is accessed by pressing the "What-If Exploration" button on the Agent Manager Page. The Exploration Agent's task is to build a visually interactive model of the user's current evaluations of an issue. This model can then be used to explore

the impact of various criteria on alternatives the user has evaluated. This is accomplished through a visual representation [Figure 6-11] of the user's utility values for each alternative, which dynamically responds to a user's changes in the criteria weights. This lets the user immediately see the effect of increasing or decreasing the relative importance

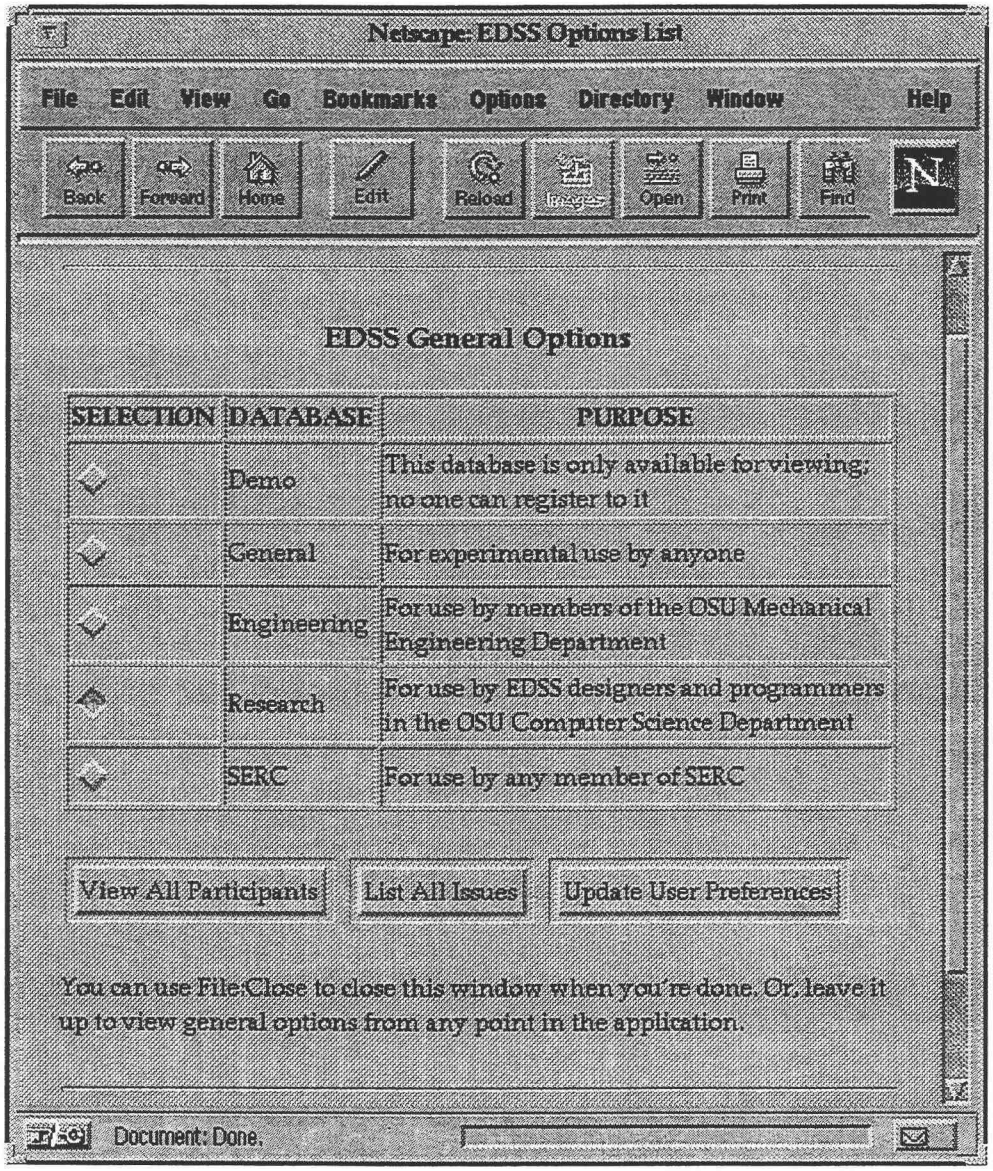


Figure 6-9: General Options Page

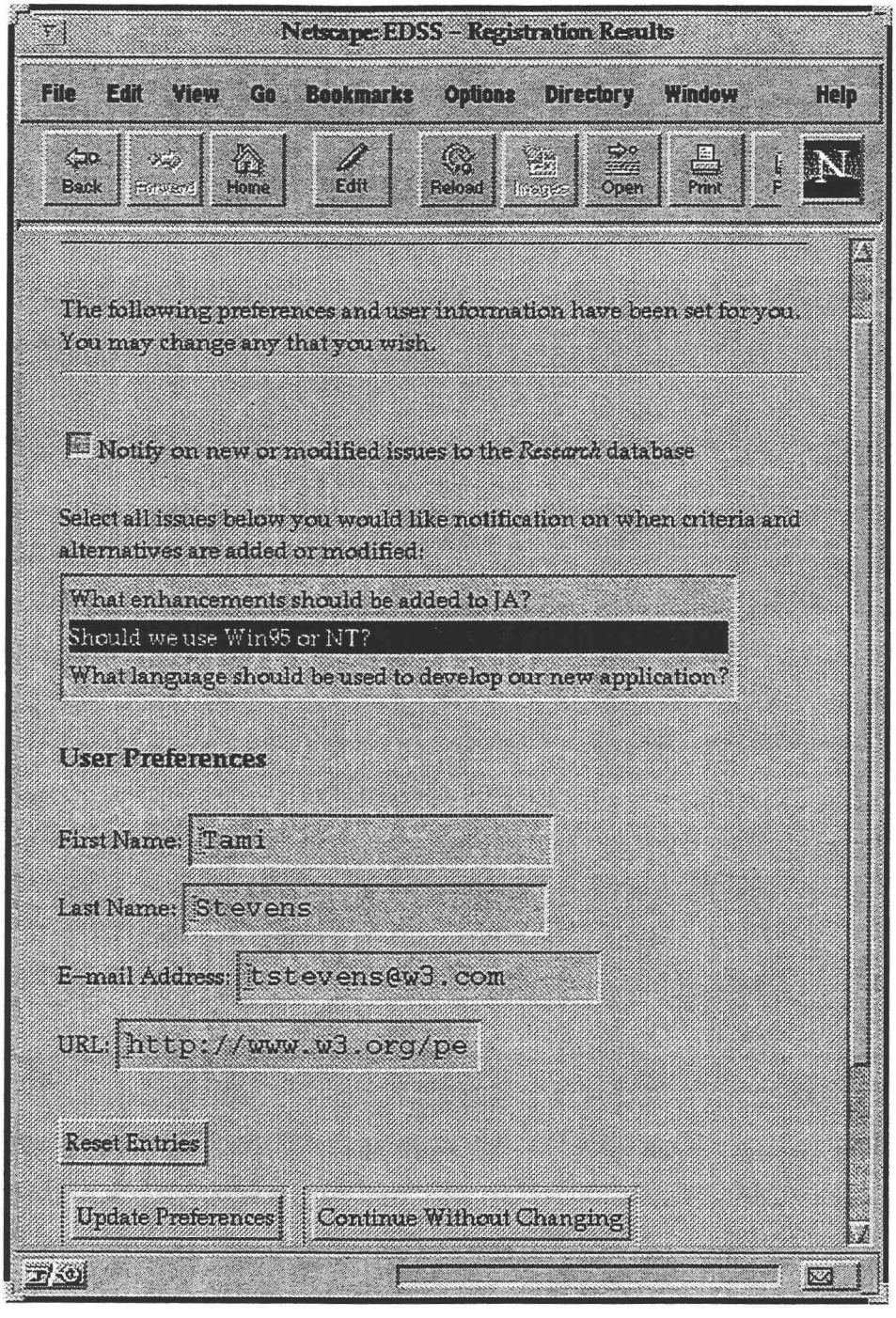


Figure 6-10: User Preferences Page

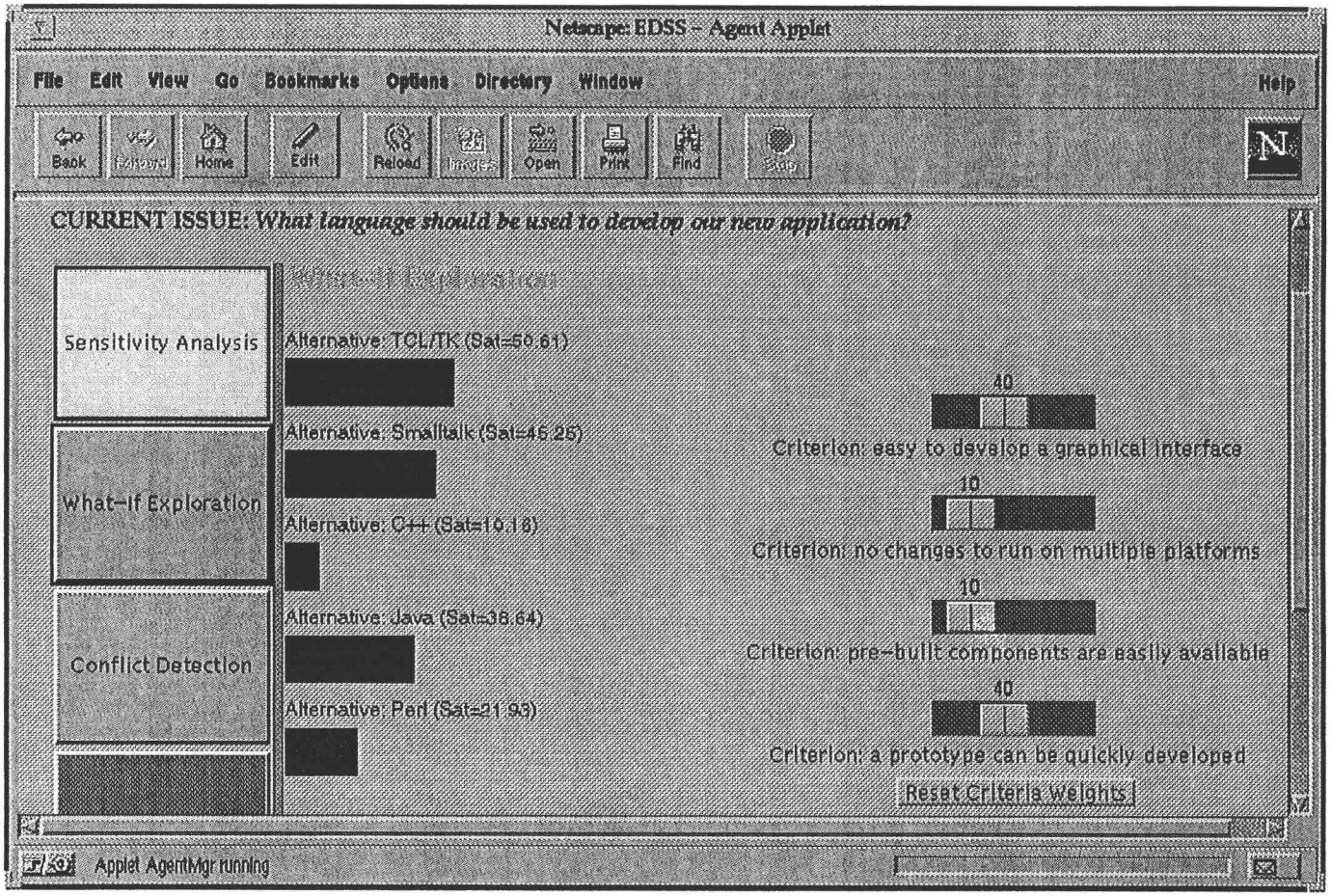


Figure 6-11: Exploration Agent

of a criterion. For example, consider the scenario presented in Figure 6-11. Based on this scenario, we can see that the first listed alternative "TCL/TK" is the most favorable one to satisfy the issue when the first and last criteria are weighted as the most important ones. However, if the user changes this by moving the sliders to give more importance to the second and third criteria, we can see that alternative "Smalltalk" looks more appealing now [Figure 6-12]. Group members can explore the impact of various criteria on their analysis of the alternatives through simple graphical interaction.

6.2.2.3 Interaction Support: Interaction Agent

The Interaction Agent is activated by pressing the "Search For Experts" button on the Agent Manager Page. This agent's task is to help the user learn from more experienced and knowledgeable peers. It accomplishes this by analyzing other group members' evaluations for each alternative for an issue and identifying all members who seem experienced. It presents this in a list-like form to the user [Figure 6-13]. Rather than merely presenting a textual list of names though, the agent generates some simple graphics to help the user quickly compare relative levels of experience between multiple users. This can help a user to quickly find potentially useful contacts. By clicking on any group member's name, the user can immediately see more detailed information [Figure 6-14] about the member and his/her expertise, including contact information. The agent thus gives the user a specific contact (another group member) and context (that member's expertise in a particular area) to consult.

6.2.3 General Design and Implementation Issues

6.2.3.1 Impact of Application Location

A fundamental difference between PC-EDSS and the web-based version of WEDSS 1.0 is the concept of where the application resides. PC-EDSS runs on a local

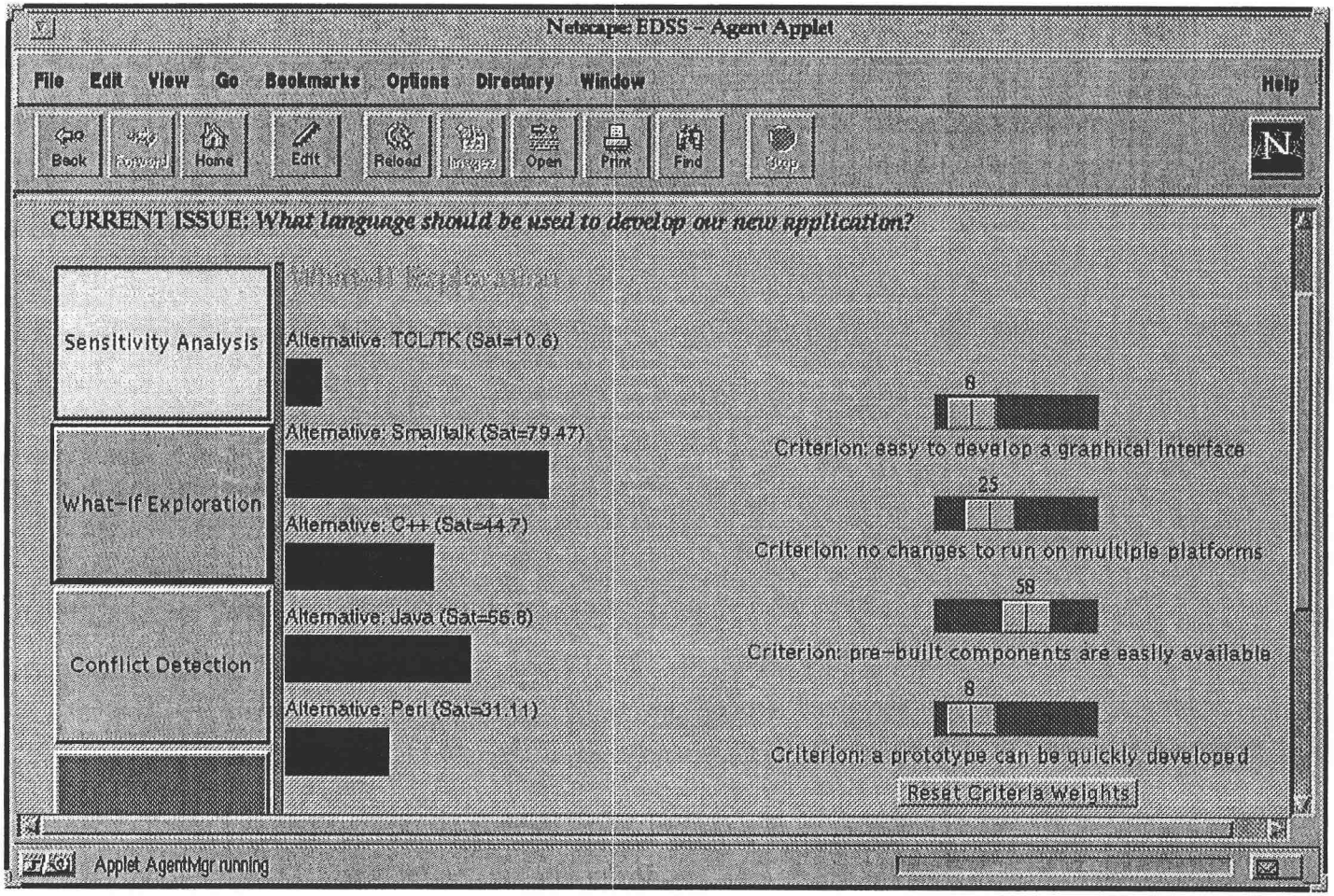


Figure 6-12: Results of Using Exploration Agent

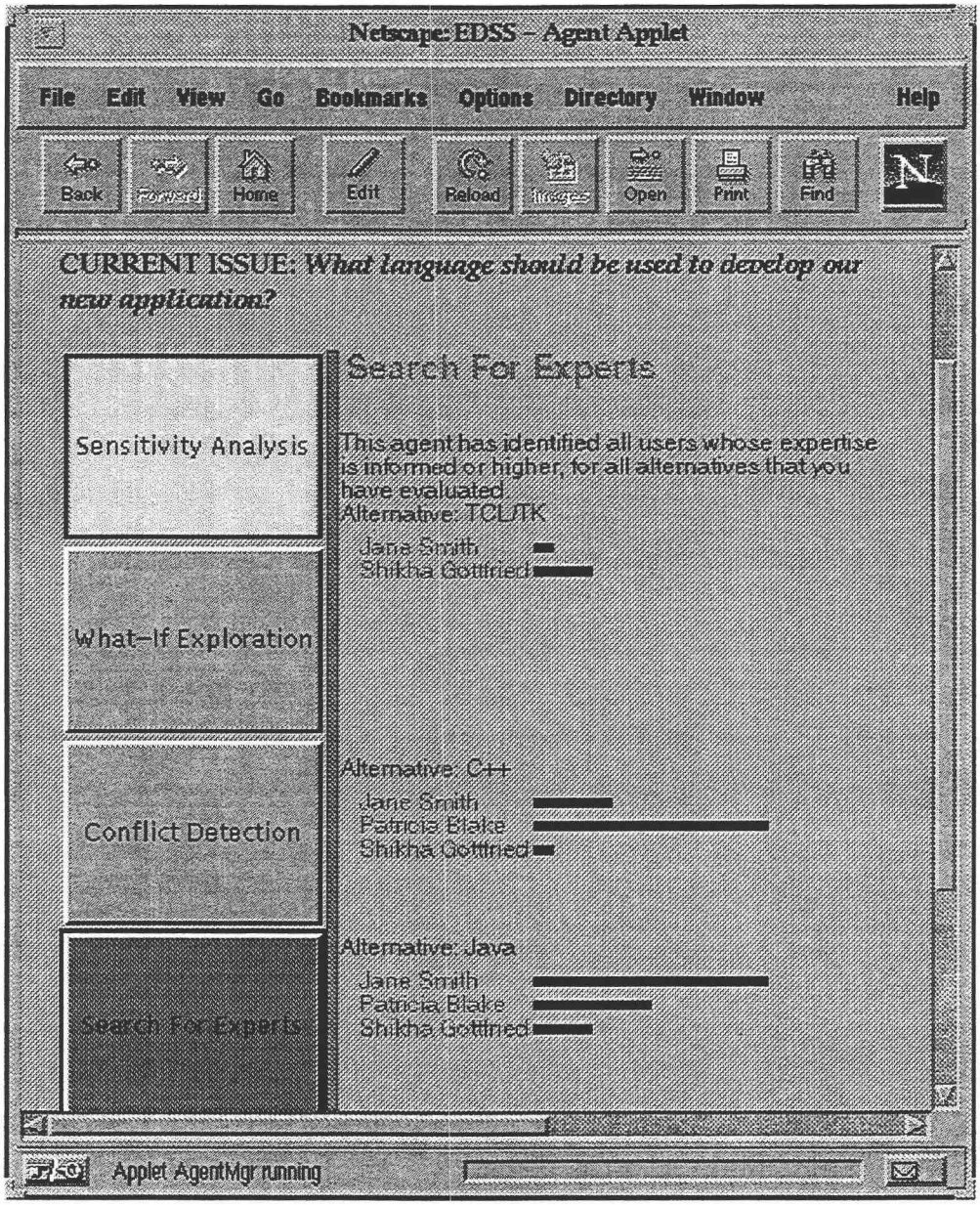


Figure 6-13: Interaction Support: "Search For Experts" Agent

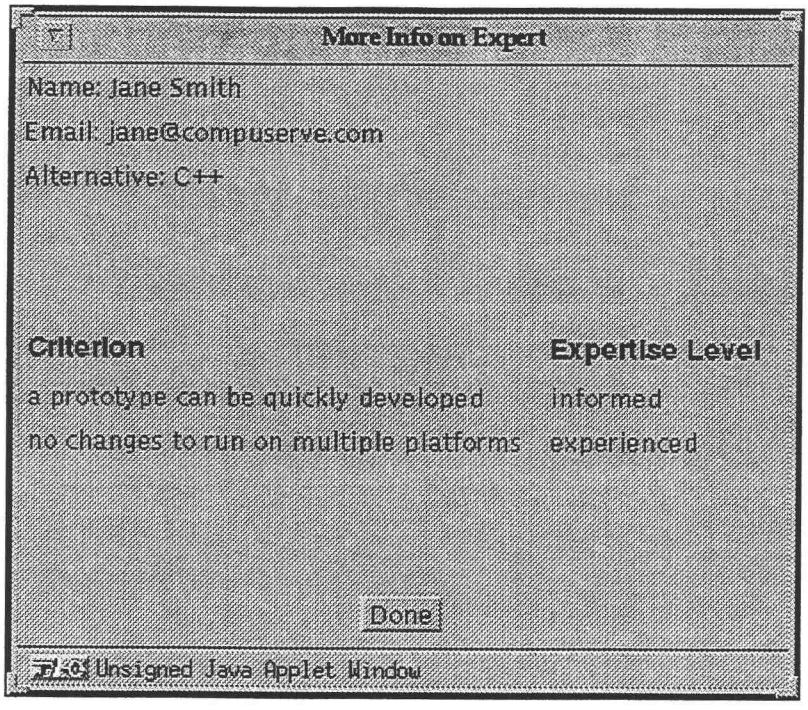


Figure 6-14: Details From "Search For Experts" Agent

machine and the entire group is expected to access it on that one machine. Multiple copies of PC-EDSS can be run by any number of different groups, each with their own set of issues. EDSS 1.0 on the other hand, runs on a simple web site that can be accessed by any member of any group, or any other interested party. Because of this, it made sense to partition the issue information by group or organization. By selecting a particular database to enter, the user will view only issue information relevant to that group or organization. Another advantage is that new databases can be easily added to WEDSS and become immediately available for users.

6.2.3.2 Language choices

Most of the web pages that comprise WEDSS are generated in response to user input which is sent to the application. CGI scripts [13] parse user choices passed from

FORM elements, execute code based on that input (often retrieving issue specific data from a group database), then output HTML code to return formatted results to the user. WEDSS uses this model to elicit user selections, then accesses the appropriate database to return issue dependent data that appears in web pages that are similar to screens in a stand-alone application.

The Agent Manager and the two agents it handles were implemented in Java. Since the database resides on the server side, the relevant information had to be sent to the client. Because the various agents use much of the same data, it made more sense to bring over all the data to the client machine rather than connecting to the server each time any data was required. Java also provides more sophisticated components than FORMS elements, allowing us to implement a more flexible interface. For example, we can support event driven actions such as those that allow the Exploration Agent to give the user immediate visual feedback.

6.2.3.3 The SQLPerl Database

All the data for WEDSS is stored in a freely available SQLPerl [47] database developed at the University of Oregon. This database is implemented in the perl language and uses SQL calls which can be easily embedded into perl scripts. Since all of WEDSS 1.0 and much of WEDSS 1.1 is implemented completely in perl, this feature allowed us to easily access data within our CGI scripts.

7. Discussion of the Prototype

We developed the prototype presented in Chapter Six as an example of one way that the ideas in our framework could be implemented. We used the framework to consider what new features could benefit web-based groups. In this chapter we discuss what motivated our particular choices and some observations we made while developing them. We also describe some of our general experiences adapting EDSS into a web-based application.

7.1 Design Motivations

7.1.1 A Consideration of Process Support

Two aspects of the functionality described by the framework for process support (section 4.1) were already part of the original version of EDSS: (1) the structure imposed by IBIS ensures that group members' ideas are captured in a structured yet informal format, and (2) the design discussion can be retrieved by reviewing each group member's evaluations. Both of these aspects combine to provide basic support for group memory. A tutorial and documentation describing the general concepts behind EDSS provide some aid (albeit minimal) to help users understand the model as well as the system. These features were adopted in WEDSS so that they could be used by web-based groups. This was done simply by making them accessible on the web. Groups can now access the same information that was previously available only on one local machine.

While web access made group work more flexible by allowing group members to participate from any location they wanted to, this introduced the problem of maintaining group awareness of the state of the discussion. In WEDSS 1.0, users could only learn of new issues, alternatives, or criteria by visiting the web site and viewing each issue one at a time. Every group member would need to monitor the progress of the design discussion.

The Notification Agent automates this monitoring, reducing the extra work that web-based groups would otherwise have to do. Notification is made through email rather than requiring group members to check a central site. Thus, group members do not need to visit the WEDSS web site unless they have work to do. This mechanism for notification also gives group members a transcript of changes in the design discussion that they can store for their own use. These features were specifically implemented to support auxiliary tasks.

Although the monitoring process is automated, the Notification Agent relies on group members to indicate their requests for notification. This is managed in WEDSS through point and click selections of topics for notification (refer to Figure 6-9), but there are still situations that could decrease the agent's effectiveness—after the initial prompting by the system during user registration, users must initiate changes themselves. Rather than expecting users to undertake this, a more intelligent agent could examine what the user looks at when the user is in the system, and extrapolate interesting events to monitor. While such technology is possible, this would be a separate research topic.

7.1.2 A Consideration of Analysis Support

Analysis support in the original version of EDSS consolidates the qualitative opinions of participating group members and provides a quantitative means of comparison between alternatives. These methods are similarly used in WEDSS. The same methods can be used by any web-based group on any problem defined within WEDSS. This type of analysis support focuses only on the final decision choices. The framework helped us consider alternative ways that analysis could inform the decision making process in WEDSS.

We were interested in exploring how analysis could foster problem understanding rather than direct problem resolution. While decision theoretic semantics provides quantifiable information, these results give no insight as to how they are influenced by various factors. We chose to deconstruct a particular aspect of the quantifiable results—the effect of criteria on individual decisions. This was motivated by the belief that web-

based users of WEDSS could benefit from a means to help them make sense of the quantitative results. This particular task is meant to provide help for the individual rather than the group, to increase the personal benefit of using WEDSS. We would hope that this would indirectly benefit the group, by making the group's members more confident of the decisions they make. The Exploration Agent was designed to achieve this through interactive event simulation, allowing the user to engage in an examination of cause and effect between criteria weights and the decisions represented by utility values.

We implemented one task to explore how sensemaking could be incorporated in the system. Although identifying the task was simplified by the guidance of the framework, we found that we had to spend considerable effort determining how it would interact with the user. Unlike many process support tasks, we found no concrete examples of how sensemaking can be implemented. The Exploration Agent may help generate new ideas for real support of sensemaking.

7.1.3 A Consideration of Interaction Support

Unlike process and analysis support, there is no explicit interaction support in the original version of EDSS. However, groups using EDSS can communicate off-line to exchange any kind of information. This specifically allows them to assess the expertise levels of various group members based on their contributions to the discussion, or talk to the group to identify experts for context specific information.

We assume that many remote groups may not know each other well enough to be familiar with every member's areas of expertise for each particular issue. Since members in web-based groups contribute to the group discussion independently they do not have the same opportunities as traditional groups to casually acquire this information in the course of their work. This puts them at a disadvantage when they need to find other group members they can turn to for help. The exchange of knowledge that takes place between "experts" and "novices" in a group was identified as a specific group gain ("learning") in field studies of group interaction [36]. We developed the Interaction Agent to restore this gain to web-based groups.

The results of good interaction seem less tangible than those from effective process or analysis. It was difficult to first consider the need for interaction support, because no one seemed to identify its benefits. Discussion with the EDSS research group elicited interest in process and analysis support but interaction seemed to be something that would be handled by the users. We think the reason for this stems from a view that promoting interaction simply means providing a communication mechanism between group members. The incentive to interact with others in this situation must come from the individuals within each group. The framework introduced a different view of supporting interaction, one that advocated guiding users to encourage interaction and establishing an environment conducive to interaction. Developing a specific idea for interaction support was much easier once we focused on supporting the specific group gain of 'learning'. We suspect that if specific interaction goals can be identified, then tasks to support them can be designed more easily.

7.2 Experiences Developing a Web-based Application

The number of web-based applications are increasing, but they are still a new enough phenomenon that there is not much published information about related development issues and experiences. In the following section, we share some of our implementation experiences encountered while developing WEDSS.

7.2.1 Developing a Graphical Interface for WEDSS

Two of the most common ways to support a web-based application with a graphical user interface are with the use of either Java applets embedded into web pages or HTML FORM elements and CGI scripts. Our prototype was developed using a combination of both, providing an interesting opportunity for comparison.

7.2.1.1 Interface Development Using FORM Elements

The interface to WEDSS 1.0 was written entirely with HTML FORMs, primarily because it represented the best way of creating an interactive application at the time we undertook development and because the underlying CGI scripts could be rapidly developed in Perl. The use of FORM elements such as radio buttons and text boxes presented a dynamic change from purely text based static pages. User input could be captured and used to generate context dependent web pages similar to screens in a standalone application. While this represented a major advance in terms of web interactivity, the look and feel of such HTML FORM elements was still quite constrained by normal GUI standards. The placement of HTML FORM elements can not be precisely controlled to specific x-y coordinates; rather, they are dependent on HTML standards which favor flexible formatting rules that give considerable control to web browsers and user preferences. For instance, the look of a radio button including its size can be subject to change from browser to browser, which can affect its placement within the interface. Interfaces require a more precise layout control, both to give users a consistent interface and to make sure that related interface items stay together. For example, a user with a ten-inch wide browser window may see three action buttons in one row, but a user with a narrower seven-inch wide browser window may see only two buttons on one line while the third button is carried to the next line. This could confuse users into thinking that only two options are available or that there are two separate groups of buttons.

Another constraint of the use of FORM elements is the relative "deadness" of the elements. Selections, such as choosing an item from a listbox, are acted upon only when a POST or GET method is sent to the receiving script which occurs only when the user presses a submit button. Unlike event-driven interfaces, where myriad events can be associated with elements to generate dynamic actions, web pages with FORM elements can only call actions to execute once per page. An example of where this forced us to create a more complicated interface is the Issues Page in WEDSS (refer to Figure 6-3). The desired functionality was to display the more detailed descriptions for each issue alongside the listing of the issues, changing the description as a user highlighted a different issue. However, because a listbox selection is not recognized until a submit button is

pressed, there was no way to dynamically change the description. Our solution was to take the user to a new web page showing the description of the issue. While we were able to provide the information that we wanted to convey to the user, this workaround made the overall functionality of the interface more unwieldy by introducing yet another web page that the user had to navigate. Overall, FORM elements limit the sophistication of the interface of web-based applications. However, they are fairly easy to develop along with the CGI scripts that drive them, making the combination an effective tool for rapid prototyping—a consideration that can be quite important since the area of web-based work is developing rapidly.

7.2.1.2 Interface Development using Java

Most of the development of WEDSS 1.1 was implemented in Java. There were several reasons behind this change in language: (1) the functionality we were seeking, such as multiple agents executing at one time and support for dynamic graphics, could be supported in Java; (2) Java provided greater layout control for the graphical user interface; and (3) since this was a prototype experiment, this gave us the opportunity to explore web-based application development using Java, which had become popular for constructing web-based applications. We kept the original Perl version of WEDSS 1.0 so that we could concentrate on implementing the new ideas for agents that had emerged from our research in developing the framework. The old and new work on WEDSS were easily integrated by embedding an applet into a web page generated by a Perl script.

Java proved to be a good choice for displaying the support handled by the agents since its libraries provide a much richer set of widgets as well as more precise layout control. Java's graphics support also allowed us to create dynamic images that the user could interact with (such as sliders), rather than being limited to a pre-defined set of FORM elements. The flexibility offered by Java is offset by the development time to use it. Although the WEDSS 1.0 and WEDSS 1.1 interfaces had different functionality it seemed that more code and components were required to work with Java. Widespread development tools for Java are scarce (although some have appeared in the last six months) and finding ready-made components outside of the provided class libraries

required a fair amount of search through the internet. As Java becomes more widespread, we anticipate that a richer support structure will decrease development time and effort.

7.2.2 Performance Issues

The critical components affecting performance of WEDSS are the calls to the server to retrieve a script to generate a web page in WEDSS. Since these scripts create new web pages each time they are called, they can not take advantage of any browser caching mechanisms, which usually speed up web page display. Consequently, long distances and slower connections to the web server that WEDSS resides on will affect user access to any of the WEDSS web pages. While performance issues were not a focus in this version of the prototype, preliminary experiences with the system suggest this issue will require attention for any future production level releases. We are investigating a re-implementation of WEDSS completely in Java, to take advantage of transferring much of the application's work to the client-side machine rather than overloading the web server at the WEDSS site.

Both CGI scripts and Java applets offer opportunities for creating successful web-based applications. In our experiences, Perl scripts are well suited for rapidly prototyping most ideas, but production level applications will benefit more from the effort and time required to develop sophisticated Java applets.

8. Future Work

There are many possibilities for further research to extend the framework and the prototype.

8.1 New Considerations for the Framework

One goal of our work was to identify the kinds of support that web-based design groups would need in order to collaborate effectively. The framework is the result of our research. In this section we identify potential directions to pursue to build on this work.

8.1.1 Refining the Framework

We found valuable ideas about group interaction by examining group process theory research. This suggests that disciplines unrelated to computer science may yield potentially interesting models of group work that could give new perspectives to application developers. Psychology and sociology are two areas that include study of how groups function and that may offer new insights to refine the concepts described by the framework.

Although in this thesis we have listed specific ideas for each type of support, these are not meant to suggest a complete list. Our own experiences developing the prototype using the framework showed us that these detailed descriptions were quite valuable in clarifying the role of each type of support and in suggesting ideas that we could implement. Thus we think further enumeration of tasks would be a useful area for further work.

Our understanding of the way web-based groups function is based on observations and limited research because there are few published papers that discuss this topic. This whole area of web research is very new since the web itself is relatively young (1991) and applications that run on the web have appeared only in the last year or so. Several web-

based collaborative applications have recently emerged and we anticipate that research and experience reports from these as well as other new work will help us refine our framework.

8.1.2 Validating the Framework

Our prototype validated the idea that we could build an application based on the support defined by the framework. Building more applications with the framework will help to further strengthen this claim. Application development by other developers using this framework will also make this claim more objective. The value of the framework to application developers will be determined by eliciting feedback from developers about their perception of its influence on the design of their applications. The success of these applications in supporting group work will help us evaluate how well the framework describes the proper types of support. Care will need to be taken to ensure a way to evaluate the framework's effectiveness separate from the effectiveness of any particular implementation.

8.1.3 Applicability Beyond Engineering Design

There are several concepts suggested by this framework that seem applicable to more general group resolution problems than engineering design alone. Process and interaction support seem particularly adaptable to any type of group collaboration effort. It would be interesting to test whether an application such as WEDSS could be used for other casual decision making (we suspect it can), and also whether the framework could be used to influence the design of applications for non-engineering group design. This could involve investigating which types of support to forego and whether different functional requirements would need other types of support not identified in our framework. This direction of research could yield interesting opportunities for examining how well the framework can be generalized.

8.2 Extensions to the Prototype

WEDSS is a prototype implemented to adapt the ongoing work in EDSS to a web-based environment. In addition it implements extensions to explore ideas of support defined by our framework. These areas offer rich sources of new ideas to enhance WEDSS.

8.2.1 Research in EDSS

Ongoing work with the PC version of EDSS includes researching other methods for alternatives evaluation, other analysis techniques, and business support strategies. Our group is currently investigating the use of measured evaluation in addition to the original method of absolute boolean evaluation. Users will be able to compare one alternative against another rather than evaluating each only on its own. Efforts to make EDSS a more industrial application include means for generating reports on issues. User studies of groups engaged in decision making using EDSS have been videotaped and analyzed and the results from this may yield observations about group use of EDSS that can be applied to WEDSS as well.

8.2.2 Robust Search and Retrieval

Decisions are best made when relevant information can be freely and easily accessed by all the decision makers. WEDSS would benefit greatly if the information easily captured by the IBIS structure could just as easily be retrieved in a flexible manner. Enhancing the information retrieval capabilities in WEDSS to support a more extensive and free form of search and retrieval would allow users to query for specific information they need. Information such as design rationales could be better utilized. A more sophisticated search and retrieval mechanism could also provide the basis for agents that initiate searches for potentially useful information and customize the results for each user.

8.2.3 Additional Agents

There are many additional agents that could be incorporated into WEDSS. A workflow agent could support business process issues, such as monitoring the level of discussion on issues, helping the group to close and resolve issues when deadlines come up, and sending out summary reports as required. Various analysis agents could be employed to model decision theoretic techniques, such as sensitivity analysis or concept maps that graphically model the state of an issue. Alternatively, agents could support more sensemaking activities such as triangulation. An example of an agent supporting this sensemaking strategy is one that compares evaluations to check the consistency of opinions. The challenge of bringing group members together will also require more work. Agents may need to act as mediators or facilitators to encourage group interaction. A negotiation agent could be responsible for helping group members resolve differences of opinion. In summary, the opportunities for agents abound; the challenge is to find tasks that are well-suited to manipulation by agents and that will also be of benefit to users.

8.2.4 A Complete Java Implementation

The performance and interface issues that arose in WEDSS 1.0 led us to implement part of WEDSS 1.1 in Java. Research is currently underway to reimplement the design of WEDSS using more components in Java. This includes a database in Java to replace the current SQLPerl database, and a WEDSS application applet to replace the CGI Perl scripts that drive the IBIS model in the current version of the application.

9. Conclusion

Collaborative applications have not been as successful as expected, given the effort invested to build them. The cost to adopt and maintain them has prevented widespread use among small distributed groups and they have often required too much extra effort by groups members. Application developers have had a poor understanding of the extra work that collaborative applications add to group collaborative efforts and thus have often not included any means to reduce this work or compensate for it by increasing the direct benefit to group members and their groups.

We believe the web can make collaborative applications more accessible and can be an effective medium if the asynchronous and distributed nature of work is supported. We have defined a framework to describe the three types of support that will facilitate work by web-based groups engaged in engineering design. The purpose of this framework is to help developers become more aware of the types of support to consider so that they can incorporate explicit ways of facilitating group efforts in collaborative applications. We identified three types of support—process, analysis, and interaction—by examining literature in CSCW, group process support and decision support and integrating the ideas found in these areas with the particular goal of supporting web-based design. We developed a prototype of a web-based application to examine this use of the framework and found that it helped us design explicit support for web-based groups.

The contribution of our research is a framework that describes ideas that can influence application development by helping developers become more aware of how groups should be supported. We have specifically identified ways to facilitate web-based group work, an area that is still very new.

Bibliography

[1] Ackerman, M. S., "Augmenting the Organizational Memory: A Field Study of Answer Garden", In *Proceedings of CSCW 94, Conference on Computer-Supported Cooperative Work*, (October 1994), pp. 243-252.

[2] Atwood, M. E., B. Burns, D. Gairing, A. Girgensohn, A. Lee, T.Truner, S. Alteras-Webb, and B. Zimmerman, "Facilitating Communication in Software Development", In *Proceedings of DIS 95, Designing Interactive Systems: Processes, Practices, Methods and Technologies* (August 1995), pp. 65-73.

[3] Balasubramaniam, R., and Sengupta, K., "Multimedia in a design rationale decision support system", *Journal of Decision Support Systems,* Vol. 15 (1995), pp. 197-210.

[4] Bellotti, V., S. Buckingham Shum, A. MacLean, and N. Hammond, "Multidisciplinary Modeling in HCI Design... In Theory and In Practice", In *Proceedings of CHI 95,* (May 1995), pp. 146-153.

[5] Berners-Lee, T., "WorldWide Web Seminar", *W3C* (April 1996), Online, Internet, Available at http://www.w3.org/pub/www/Talks/General.html

[6] Bikson, T. K., "Notes From the Conference Chair", In *Proceedings of CSCW 90, Conference on Computer-Supported Cooperative Work*, (October 1990), pp. iii-v.

[7] Blanning, R. W., and D.R. King., *Current Research in Decision Support Technology,* IEEE Computer Society Press, CA, 1992, pp.163-165.

[8] Boutell, T., "WWW FAQ", (April 1996), Online, Internet, Available at http://www.boutell.com/faq

[9] "BrainWeb", *Delft University of Technology* (October 1995), Online, Internet, Available at http://www.sepa.tudelft.nl/~gdr/brainweb/bw0.html

[10] "BSCW - Basic Support For Cooperative Work", *BSCW Home Page* (November 1996), Online, Internet, Available at http://www.bscw.gmd.de/BSCW2-Intro

[11] Bui, T. X., *Co-oP: A Group Decision Support System for Cooperative Multiple Criteria Group Decision Making*, Springer-Verlag, New York, 1987.

[12] Bullen, C. V., and J. L. Bennett, "Learning From User Experience with Groupware", In *Proceedings of CSCW 90, Conference on Computer-Supported Cooperative Work,* (October 1990), pp. 291-302.

[13] "Common Gateway Interface", *NCSA* (March 1996), Online, Internet, Available at http://www.hoohoo.ncsa.uiuc.edu/cgi/intro.html

[14] D'Ambrosio, B., and D. Ullman, "Decision Problem Representations for Collaborative Design", In seminar *at International Joint Conference on Artificial Intelligence* (July 1995), Online, Internet, Available at http://www.cs.orst.edu/~dambrosi/ijcaibmp.ps.z

[15] Ellis, C. & Wainer, J., "Goal-based models of collaboration", *Collaborative Computing*, Vol. 1 (1994), Online, Internet, Available at http://hermes.chaphall.co.uk/cc.html.

[16] "Facilitate.com Product Description", *C.A. Facilitator* (November 1996), Online, Internet, Available at http://www.facilitate.com/CAF/web.html

[17] Fowler, J., D. Baker, R. Dargahi, V. Kouramajian, H. Gilson, K. Brook Long, C. Petermann, and G. A. Gorry, "Experience with the Virtual Notebook System: Abstraction in Hypertext", In *Proceedings of CSCW 94, Conference on Computer-Supported Cooperative Work* (October 1994), pp. 133-143.

[18] Frivold, T. J., R. E. Lang, and M. W. Fong, "Extending the WWW for Synchronous Collaboration", In *Electronic Proceedings of 2nd International World Wide Web Conference* (September 1994), Online, Internet, Available at http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/CSCW/frivold/frivold.html

[19] Goldberg, Y., M.Safran, and E. Shapiro, "Active Mail - A Framework for Implementing Groupware", In *Proceedings of CSCW 92, Conference on Computer-Supported Cooperative Work* (October 1992), pp. 75-83.

[20] Grudin, J., "CSCW: History and Focus", (February 1995), Online, Internet, Available at http://www.ics.uci.edu/~grudin/cscw.html

[21] Grudin, J., "Eight Challenges For Developers", *Communications of the ACM* (January 1994), pp. 93-105.

[22] Grudin, J., "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces", *In Proceedings of CSCW 88, Conference on Computer-Supported Cooperative Work* (October 1988), pp. 85-93.

[23] Herling, D., D. G. Ullman, and B. D'Ambrosio, "Engineering Decision Support System", Oregon State University Mechanical Engineering Department (May 1995).

[24] Hoffer, J. A., and J. S. Valacich, "Group Memory in Group Support Systems: A Foundation for Design", *Group Support Systems: New Perspectives*, L. M. Jessup and J. S. Valacich, Eds., Macmillan Publishing Co., New York, 1993, pp. 214-229.

[25] "HTML: Working and Background Materials", *W3C* (November 1996), Online, Internet, Available at http://www.w3.org/pub/www/MarkUp

[26] "Internet Collaboration Starting with Collabra Share", *Netscape Communications* (September 1995), Online, Internet, Available at http://www.netscape.com/comprod/announce/collabra.html

[27] Jessup, L. M., "Profiles of Innovative Groupware Uses", *Groupware Central* (April 1996), Online, Internet, Available at http://ezinfo.ucs.indiana.edu/~ljessup/gwcent3.html

[28] Jessup, L. M., and Valacich, J. S., Ed., *Group Support Systems: New Perspectives*, Macmillan Publishing Co., New York, 1993.

[29] Kunz, W., and H. W. Rittel, "Issues as Elements of Information Systems", Working Paper No. 131, Institute of Urban and Regional Development, University of California at Berkeley, 1970.

[30] Lee, Jintae, "SIBYL: A Tool for Managing Group Decision Rationale", *In Proceedings of CSCW 90, Conference on Computer-Supported Cooperative Work,* (October 1990), pp.79-92.

[31] "Listserv Facts Sheet", *L-Soft* (August 1996), Online, Internet, Available at http://www.lsoft.com/listserv-facts.html

[32] "Lotus Notes and the Internet", *Lotus Notes* (November 1996), Online, Internet, Available http://www.lotus.com/notesr4/inter.htm

[33] "Lotus Notes: An Overview", *Lotus Corporation* (November 1996), Online, Internet, Available at http://www.lotus.com/corpcomm/22b6.htm

[34] "Majordomo FAQ", version 1.114 (November 1996), Online, Internet, Available at http://www.greatcircle.com/majordomo/FAQ

[35] "Mosaic for X version 2.0 Fill-Out Form Support", NCSA (June 1996), Online, Internet, Available at http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html

[36] Nunamaker, J.F. Jr., A. R. Dennis, J. S. Valacich, D. R. Vogel, and J. F. George, "Group Support Systems Research: Experience from the Lab and Field", *Group Support Systems: New Perspectives*, L. M. Jessup and J. S. Valacich, Eds., Macmillan Publishing Co., New York, 1993, pp. 125-145.

[37] Orlikowski, W. J., "Learning from Notes: Organizational Issues in Groupware Implementation", In *Proceedings of CSCW 92, Conference on Computer-Supported Cooperative Work*, (October 1992), pp. 362-369.

[38] Preece, J., Y. Rogers, H. Sharp, D. Benyon, S. Hooland, and T. Carey, *Human-Computer Interaction*, Addison-Wesley Publishing, California, 1994.

[39] Reder, S., and R. G. Schwab, "The Temporal Structure of Cooperative Activity", In *Proceedings of CSCW 90, Conference on Computer-Supported Cooperative Work*, (October 1990), pp. 303-316.

[40] Rein, G. L., and C. A. Ellis, "rIBIS: a real-time group hypertext system", *International Journal of Man-Machine Studies* (1991), pp. 349-367.

[41] Rhyne, J. & Wolf, C., "Tools For Supporting the Collaborative Process", In *Proceedings of UIST'92, Conference on Users, Interfaces, Software, and Technology* (November 1992), pp. 161-170.

[42] Rice, J., A. Farquhar, P. Piernot, and T. Gruber, "Using the Web Instead of a Window System", In *Proceedings of CHI 96, Conference on Human Computer Interaction* (April 1996), pp. 103-110.

[43] Rittel, H. W., and M. M. Webber, "Dilemmas in a General Theory of Planning", *Policy Sciences*, Vol. 4 (1973), pp. 155-169.

[44] Rodden, T., "A Survey of CSCW Systems", *Interacting With Computers*, Vol. 3 (1991), Online, Internet, Available at ftp://ftp/comp.lancs.ac.uk/pub/reports/1992/cscw.13.92.ps.z

[45] Savage, L. J., *The Foundations of Statistics, 2nd ed.*, Dover Publications, New York, NY, 1972.

[46] Schneiderman, B., *Designing the User Interface,* 2nd ed., Addison Wesley Publishing Company, Menlo Park, CA, 1992.

[47] "SQLPerl Database Home Page", *University of Oregon* (February 1996), Online, Internet, Available at http://www.cs.uoregon.edu/research/se/hoekstra/sqlperl/index.shtml

[48] "TCBWorks: Webware for Teams", *Terry College of Business, University of Georgia* (March 1996), Online, Internet, Available at http://tcbworks.cba.uga.edu

[49] Thistlewaite, P., and S. Ball, "Active FORMs", In *Proceedings of 5th International World Wide Web Conference* (May 1996), Online, Available at http://www5conf.inria.fr/fich_html/papers/p40/Overview.html

[50] Ullman, D., *The Mechanical Design Process*, McGraw-Hill, Inc., San Francisco, CA, 1992.

[51] Ullman, D., "Issues Critical to the Development of Design History, Design Rationale, and Design Intent Systems", *Design Theory and Methodology 94*, DE-Vol. 68 (1994), pp. 249-258.

[52] Ullman, D., and B. D'Ambrosio, "Taxonomy for classifying engineering decision problems and support systems", *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* (1995), pp. 427-438.

[53] Vogel, D. R., J. F. Nunamaker, Jr., W. B. Martz, Jr., R. Grohowski, and C. McGoff, "Electronic Meeting System Experience at IBM", in *Current Research in Decision Support Technology*, R. W. Blanning and D.R. King, Eds., IEEE Computer Society Press, California, 1992, pp. 158-176.

[54] "W3C-The World Wide Web Consortium", *W3C* (November 1996), Online, Internet, Available at http://www.w3.org

[55] Wagner, C., "Facilitating space-time differences, group heterogeneity and multi-sensory task work through a multimedia supported group decision system", *Journal of Decision Support Systems*, Vol. 15 (1995), pp. 197-209.

[56] *Webster's 9th New Collegiate Dictionary*, Merriam-Webster Inc., Springfield, Massachusetts, 1983.

[57] Weick, K. E., and D. K. Meader, "Sensemaking and Group Support Systems", *Group Support Systems: New Perspectives*, L. M. Jessup and J. S. Valacich, Eds., Macmillan Publishing Co., New York (1993) pp. 230-252.

[58] Woo, T. K., and M. J. Rees, "A Synchronous Collaboration Tool for World Wide Web", In *Electronic Proceedings of 2nd International World Wide Web Conference* (October 1994), Online, Internet, Available at http://141.142.3.70/SDG/IT94/Proceedings/CSCW/rees/SynColTol/html

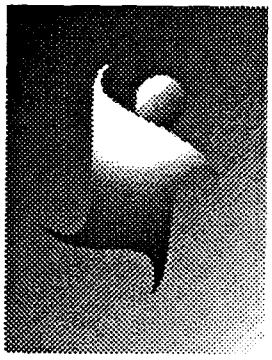
[59] Yakemovic, K.C. Burgess, and E. J. Conklin, "Report on a Development Project Use of an Issue-Based Information System", In *Proceedings of CSCW 90, Conference on Computer-Supported Cooperative Work* (October 1990), pp. 1-13.

Appendices

Appendix A: EDSS Web Site

URL: http://www.cs.orst.edu/~dambrosi/edss/info.html

Engineering Decision Support System Info Site



About EDSS | EDSS Tutorial | EDSS Web Prototype | Related Papers | Project Group Members

---

About EDSS

The Engineering Decision Support System (EDSS) is a web-based system for design team decision support. The current version is 1.0, released February 1996.

- Motivation
- Decision model
- Contributions of this research
- Summary

EDSS Tutorial

We have a step by step tutorial available for EDSS that will walk you through a simple example decision problem. We highly recommend that you step through the tutorial if you are unfamiliar with EDSS. To try the tutorial, click here.

EDSS Web Prototype

Version 1.0 is currently available for use.

Related Papers

Engineering Decision Support System, by D. Herling, D. Ullman, and B. D'Ambrosio

(edssdtm1.ps.Z)

Decision Problem Representations for Collaborative Design, by B. D'Ambrosio and D. Ullman

(ijcaibmp.ps.Z)

A Taxonomy For Classifying Engineering Decision Problems And Support Systems, by D. Ullman and B. D'Ambrosio

(tax.ps.Z)

Current Project Members

- Bruce D'Ambrosio, Computer Science, OSU.
- Shikha Ghosh Gottfried, Computer Science, OSU.
- Derald Herling, Mechanical Engineering, OSU.
- David Ullman, Mechanical Engineering, OSU.

Shikha Ghosh Gottfried: gottfrsh@research.cs.orst.edu

Last updated: April 19, 1996

URL: http://www.cs.orst.edu/~dambrosi/edss/edss_mot.html

EDSS Motivation

Desiderata:

- Many crucial decisions are made early in design.
- These decisions are made based on partial, qualitative information.
- This early conceptual design largely determines the quality of final product.
- Designers work in teams.
- Teams are often geographically distributed.
- Design teams can be substantially more effective with minimal, unobtrusive support for decision making.

Goal: Distributed access to shared, active, design database.

- *Distributed access:* Available on the web.
- *Shared:* Design teams share a common, evolving view of the discussion.
- *Active:* Decision assistant records and analyzes decision structure to help team reach consensus.
- *Design:* We are focusing on early, conceptual design.
- *Decision Assistant:* Records and analyzes design information.

Back to EDSS Info Site

Forward to Decision Model

EDSS tutorial / OSU / dambrosi@research.cs.orst.edu

Last updated:February 17, 1996

URL: http://www.cs.orst.edu/~dambrosi/edss/edss_dec.html

EDSS Decision Model

An elaboration of the IBIS IPA (Issues, Proposals, Arguments) model.

- *Issue:* A problem to be resolved or decision to be made.
  e.g., *What language should we select for our new products?*
- *Proposal:* A suggestion for resolving an issue. We call these Alternatives.
  e.g., *C++*
- *Criterion*: A target or benchmark the selected alternative should meet.
  e.g., *Must have high quality development environments for Unix, PC, and Mac.*
- *Argument*: In our model, a statement that an alternative will or will not satisfy one of the criteria we have developed for evaluating alternatives.
  e.g., *Will satisfy: G++ on Unix, VC++ on Windows95, and CodeWarrior on the Mac.*

More on Arguments:

- Any team member can make an argument on any subset of the alternative/criterion pairs.
- Team members bring diverse levels of *knowledge* to the group.
- Team members may be more or less *confident* in their assesments.
- EDSS permits team members to record both their level of knowledge and confidence with each assessment.
- EDSS combines arguments to generate an overall assessment of the viability of each alternative

Back to Motivation Page

Forward to Contributions

EDSS tutorial / OSU / dambrosi@research.cs.orst.edu

Last updated: February 17, 1996 SGG

URL: http://www.cs.orst.edu/~dambrosi/edss/edss_con.html

EDSS Contributions

EDSS relies on a fairly standard multi-attribute utility model combined with a mundane independent expert model for argument fusion. So what's new?

- We have reduced this model to the core for light-weight decision support.
- Team members can comment on only those alternative/criterion pairs about which they feel strongly: complete information is not needed.
- We have included models of varying levels of team member expertise and confidence in a simple, robust decision support model.
- We have used this model to provide decision-theoretic semantics to IBIS, a proven aid for structuring group decision processes.

Back to Decision Model

Forward to Summary

EDSS tutorial / OSU / dambrosi@research.cs.orst.edu

Last updated: February 17, 1996 SGG

URL: http://www.cs.orst.edu/~dambrosi/edss/edss_sum.html

EDSS Summary

Distributed design teams benefit from the *focus*, *history*, and *active assistance* of a tool like EDSS (formal study under way).

We are exploiting research in:

- Design theory and methodology,
- Decision theory, and
- web-based collaboration support

to explore structured support for distributed design teams.

Try it, you'll like it!

Future Work:

- Additional argument models.
- Active discussion guidance through *value of information* estimation.
- Issue interrelationships.
- History.
- Usability issues: security, interactivity (Java?), process.

Back to Contributions

Back to EDSS Info Site

EDSS tutorial / OSU / dambrosi@research.cs.orst.edu

Last updated: February 17, 1996 SGG

Appendix B: EDSS Tutorial

URL: http://www.cs.orst.edu/~dambrosi/edss/tutorial.html

EDSS TUTORIAL

Welcome to this tutorial. Our purpose is to walk you through the EDSS application so that you can familiarize yourself with how everything works before you try a session on your own.

If you have never used EDSS before, please take the time to go through this tutorial before you begin entering anything in to any of the databases.

We have set up an example exclusively for this tutorial that will allow you to see how EDSS organizes the information for you to view, and what a sample analysis can yield.

Last Updated: January 22, 1996 SGG

STEP 1: Getting to EDSS

Go to the EDSS Web Prototype Page, which is the gateway into the application. Note that if you're using Netscape 1.1n or later, a new browser window will open. This will allow you to read this tutorial while exploring EDSS in the other browser window. Alternatively, you can get there by typing in the following URL:

http://www.cs.orst.edu/~dambrosi/edss/tutorial.html

Continue to next step

STEP 2: Finding the EDSS Entry Form

Follow the link named "Check in" right below the image on the EDSS main page.

This will take you to the form located lower on the page, which you will need to fill out to be granted access into EDSS.

Continue to next step OR Go back to previous step

STEP 3: The EDSS Entry Form - Database Descriptions Link

Before filling out the form, **follow the link "Database Descriptions".**

This will come in handy later if you decide to explore other databases. The table that appears gives the name of each database and a brief description of its purpose. Note that some databases have restricted use.

When you've finished looking at the Database Description, follow the link "Return to EDSS Web Prototype Page" which will take you back to the main page.

Now follow the "Check in" link again. You should now be back at the EDSS Entry Form.

Continue to next step OR Go back to previous step

STEP 4: The EDSS Entry Form - Logging In

For this tutorial, you will be logging in as Guest. The Guest account is available on most if not all of the various databases and allows you to browse without making any changes yourself to the database.

Type "Guest" into the field to the right of "First Name".

Next, skip over the "Last Name" field (make sure you don't put anything in, not even blank spaces!).

Choose the Demo database from the selection list, which will appear when you click on the raised selection list box.

Finally, click on the button "Start EDSS". If you made any mistakes, you can click on the button "Clear Entries" which will reset the form, and you can begin again.

Continue to next step OR Go back to previous step

STEP 5: Choosing An Issue

Click and hold down the Selection Menu showing the various issue available for this database. Select the last one, titled "Where should we go on our next vacation?".

You may view a more detailed description of the issue by now clicking on the Show Description button. After you are done, follow the link "Return To Issues List" to come back to the Issues page.

This time, click on the Continue button, which will take you to the next stage.

Continue to next step OR Go back to previous step

STEP 6: Evaluation Page Overview

This page contains a lot of information, so there are several steps (steps 6, 7 and 9) that will cover this one page.

This page is used to select alternative criteria pairs and evaluate their feasibility for the current issue. An Alternative is a possible solution to an issue. Criteria are constraints to an issue. By pairing Alternatives and Criteria, the user can evaluate whether she/he thinks that a given Alternative can satisfy a specified criterion. This evaluation is captured by the selection of a Knowledge level (indicating the user's expertise) and a Confidence level (indicating how well a user believes that the alternative in question satisfies the criterion). There are a couple of sections to this page.

I. The first line shows the current issue under consideration.

II. The next line gives the name of the current participant, in this case, Guest.

III. The next section contains the selection menus of all possible alternatives and criteria currently defined for the issue.

IV. The first set of radio buttons below the menus selects the Knowledge level of the user - this marks her/his level of expertise for the current Alternative/Criteria pair that she/he is evaluating.

V. The second set of radio buttons selects the Confidence level of the user - how well she/he believes that the alternative in question satisfies the selected criterion.

VI. The rationale box allows the user to enter comments that can be viewed later and helps in understanding the choice made.

VII. The Accept Values button below the rationale box is clicked by the user once she/he is ready to enter the chosen Alternative/Criterion Pair information into the database. Note that the Guest account is restricted from entering any Alternative/Criterion Pairs into the database.

VIII. The Evaluate button is used to begin calculations of the satisfaction ratings. This will be discussed later in step 9.

IX. There are several links following the separator line; each of these will be explored in the next few steps.

Take a little time to view the current list of alternatives and criteria by clicking on the selection menus. You may view any descriptions entered by clicking on the appropriate buttons. None of your selections from this page will be entered in the database, so feel free to play around (but don't click on the "Accept Values" button just yet).

Continue to next step OR Go back to previous step

STEP 7: Adding New Alternatives And Criteria

Scroll down to the bottom of the browser page. You should see a set of 6 links, with an info button to the right of them.

The first two links - Add New Alternative and Add New Criteria - will take you to forms that allow a registered user to add their own alternatives or criteria to a particular issue. This form is very similar to the one that adds an issue - a user only need to provide 2 pieces of information - a brief name that will appear in the appropriate selection menu if accepted, and a more detailed description, which is very useful for others involved in the decision process to understand exactly what the author intended.

You may view these forms by following the links. When you are done on each form, you can follow the "Cancel" link back to this Evaluation Page.

Continue to next step OR Go back to previous step

STEP 8: Criteria Weights

Follow the link named "Criteria Weighting" in the menu list at the bottom of the page. This will take you to the Criteria Weightings Page. This page allows the current participant to define relative importance among the various criteria for an issue. Viewing the chosen weights of other participants can be done via the General Options (covered later in step 10).

When you are done looking at this page, please return to the Evaluation Page by following the Cancel... link.

Continue to next step OR Go back to previous step

STEP 9: View Summary

Follow the link named "View Summary" in the menu list at the bottom of the Evaluation Page. This will take you to the Summary Page. This page is particularly useful for getting a view of which Alternative/Criterion pairs you have already made decisions on, as well as showing you exactly what you entered. Note that as Guest, there are no pairs entered. Now click on the selection menu to open up the list of users and choose the user "Jane Tester". Once that name appears in the box, click on the Show Summary Button. This will

open up a new window (if you are using Netscape 1.1n or greater) and show the summary table for Jane Tester. Note that several of the table cells are filled in. You can quickly focus on the AC pairs that have been entered by looking for the blue check marks.
Feel free to view the summaries for the other users. All you need to do to do this is select a different user back on the other page (no need to close this page, but don't iconify it), and click the button again.
When you've finished viewing summaries, use the File menu at the top of your browser and close the window titled "Other Participant Summaries". Next use the "Go Back..." link on the window titled "EDSS-Summary" which will take you back to the Evaluation Page.

Continue to next step OR Go back to previous step

STEP 10: General Options Overview

Any time you see the following icon: [Image Not Loaded] you can click on it to take you to the General Options page, which allows you to view various information about the databases, such as the participants list, or what the other issues are in various databases. Click on this icon now to go to the General Options Page. This will open a new browser window so you won't lose your place in EDSS in the window you were just in.
This page is fairly self-explanatory. Select a database, then click on one of the buttons to either view all the registered users in that database (choose View All Participants) or to show all the issues in a particular database (choose List All Issues).When you are done, use File-Close from the browser menu to get rid of the new window.

Continue to next step OR Go back to previous step

STEP 11: Satisfaction Ratings

You are now ready to try the "Evaluate" button. Clicking on this button will start the calculations that will come up with satifactions ratings based on what all the users have contributed toward a particular issue. Click on the Evaluate button now.

The page that will appear shows a table listing the alternatives on each row. There are two columns:

1) Personal Sat, which shows the results of what the current participant's ratings are.(This is based on the Knowledge and Confidence levels entered for Alternative/Criterion pairs with each criterion is weighted with values entered by the user).

2) Overall Sat, which takes input entered by ALL of the users in the database and calculates what the group satisfaction rating is for each alternative.

In examining these tables, note that based on the values of the Alternative/Criterion pairs for Guest, the alternative listing "New Orleans" comes out the most favored (but not by much). In the Overall Sat column, note however that "Santa Fe" is the group favorite. By comparing satisfaction ratings and reviewing participant summaries, you can get a good idea of which alternatives are more favored by the group and by each person.

Go back to previous step

END of Tutorial

This concludes the tutorial for EDSS. We hope this has helped you to understand the application and learn to use it effectively. If you have any questions, comments or bugs on this tutorial, please send us email.

Prof. Bruce D'Ambrosio: dambrosi@research.cs.orst.edu

Shikha Ghosh Gottfried: gottfrsh@research.cs.orst.edu

Return to EDSS Web Information Site

Appendix C: WEDSS Code

C.1 Perl CGI Scripts

The source code can be viewed online at: http://www.cs.orst.edu/~gottfrsh/thesis/perl.

```
agent-notify.pl*
edss-ac.pl*
edss-addaltform.pl*
edss-addcritform.pl*
edss-addissue.pl*
edss-addwts.pl*
edss-agent.pl*
edss-eval.pl*
edss-modaltform.pl*
edss-modcritform.pl*
edss-modissue.pl*
edss-ops.pl*
edss-otherac.pl*
edss-othersum.pl*
edss-register.pl*
edss-showalts.pl*
edss-showcrits.pl*
edss-start.pl*
edss-summary.pl*
edss-test.pl*
edss-userprefs.pl*
edss-utils.pl*
edss-viewops.pl*
edss-weights.pl*
postdb.pl*
```

C.2 Java Source Code

The source code can be viewed online at: http://www.cs.orst.edu/~gottfrsh/thesis/java.

```
Agent.java
AgentCanvas.java
```

AgentMgr.java
BarObj.java
CDAgent.java
Chart.java
ControlPanel.java
CriterionSlider.java
DBCon.java
DataObj.java
EGAgent.java
EVAgent.java
ExpertInfo.java
ExpertObj.java
ExpertWin.java
GKAgent.java
IACObj.java
KCObj.java
SatObj.java
TableData.java
UserObj.java
VOIAgent.java
WtObj.java