Adaptable Software Agent for Retrieval and Presentation of Information in Collaborative Engineering Environments

By

Wissam Youssef Ali-Ahmad

B.E. (1991), M.E. (1995), Computer and Communications Engineering,
American University of Beirut

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Science

At the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1997

Signature of Author .....
Department of Civil and Environmental Engineering
August 18, 1997

Certified By.....
Feniosky Peña-Mora
Assistant Professor of Civil and Environmental Engineering
Thesis Advisor

Accepted By .....
Joseph M. Sussman
Chairman, Departmental Committee on Graduate Studies

Adaptable Software Agent for Retrieval and Presentation of Information in Collaborative Engineering Environments

By

Wissam Y Ali-Ahmad

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Technology

ABSTRACT

Design and development of large-scale engineering projects involve participants with various interests and expertise collaborating to produce an artifact. Such diversity is believed to cause conflicts during the decision making process; part of the problem is due to misrepresentation and miscommunication of information between peers with dissimilar preferences, the other part due to incompatible implementation of different objectives. This thesis presents an approach to decrease the problem of misrepresentation and miscommunication by using a framework: 1) for capturing knowledge about users, their preferences and the problems they tackle; and 2) for adapting the presentation of relevant information to those users. The proposed framework focuses on the issues of relevancy and adaptation within the context of information being retrieved by users. Based on the functional observations from this framework, a methodology and a system called ANGELO was built. ANGELO achieves user modeling by pro-actively observing user interactions with distinct project information. It then uses this model to adapt the presentation of relevant information according to its interpretation of the user's interests. This methodology and system combine the use of agent-based learning with information-retrieval techniques. Such methodology and system acts as a support tool for collaborative meeting environments where users need relevant presentation and recommendation of information, in order to avoid or resolve conflicts.

Thesis Supervisor: Feniosky Peña-Mora.
Title: Assistant Professor of Civil Engineering.

TABLE OF CONTENTS

LIST OF FIGURES & TABLES

ACKNOWLEDGMENTS

All my thankfulness and gratefulness goes first to almighty Allah (God), for it is without His mercy and help we couldn't have accomplished any of the achievements, and gained any knowledge.

"...Then remember Allah, as He has taught you what you did not know"

Qur'an (Chapter 2. Verse 239)

With lots of warm feelings and love, I thank my parents, father and mother, who supported me and guided me through all my life. I thank my brother Walid for always being there for me and my sister Noha for all her understanding. I am also very grateful to my wife Noura, who had patience, love and support all along my graduate hard working days at MIT. I thank our baby son Anwar for being my inspiration and providing the warm love when I needed it. In my extended family, I also thank my mother-in-law for all her motivational advises along the way.

My special thanks goes to my advisor Prof. Peña-Mora for offering the necessary guidance, and for giving me an opportunity to work on innovative research. I thank also Prof. Steve Lerman for his advice and support with my early graduate studies. I am grateful to Prof. Alex Pentland, Prof. John Williams, Dr. Henry Lieberman, and Prof. Pattie Maes for the genuine research conversations and inspirations they offered. Many thanks also go to the faculty members at the Media Lab who enlightened me with innovative and research oriented courses which helped me in my graduate studies.

I thank my colleagues at the Intelligent Engineering Systems Laboratory; Karim Hussein, and Lucio Soibelman for their help and technical guidance. I thank my friends at the Center for Educational Computing Initiatives; Jud Harward and Issam Bazzi for their continuous support and friendship. Finally, my warm thanks to my friend Dr. Mohamad Akra for his advice and guidance when I needed it.

Wissam Ali-Ahmad

GLOSSARY

Adaptation. The capability given to a system to be able to learn new rules that reflect actions to be taken in an environment.

Agent. a computational system, that has long-lived states, with goals that act from sensors and effectors. It is autonomous, therefore giving it the capability to decide which actions to take in its current situation to maximize progress and outcome.

Information Filtering. A system that helps the user by eliminating the irrelevant information and by bringing the relevant to the user's attention.

Information Retrieval. A system that retrieves documents from an indexed database requested by the user using a query of terms or boolean logic.

Programming By Demonstration. Agents that get instructed by the user through a visual programming or macro recording approach.

Software agents. A particular type of agents, inhabiting computers and networks, assisting users with computer-based tasks. Most of these agents co-exist as a front end to software applications that users need help either automating repetitive tasks or filtering information.

User Modeling. A system component that records information about the user in a profile for later usage by the system to adapt some feature or functionality.

1 Introduction

In large-scale engineering projects, the collaboration of experts in different specialties is an important characteristic of the design process and/or project development. Since these specialists have different interests and perspectives, conflict is inevitable be it over design process, product change or simple miscommunication. These conflicts, if not resolved early, create projects that are more costly, take longer to design and manufacture, and are functionally sub-optimal.

1.1 Example Scenario

In order to clarify and understand the problem at hand, an example scenario of a construction project will be described. The example is meant to depict the project elements, and interactions that originates conflicts among peers in that project.

> The project involves the construction of a building in Sydney, Australia. The team is composed of an architecture (AR), a structural engineer (SE), and a contracting company (CC). The architect sketches initial designs of the building based on site location and customer's specifications. The structural engineer gives his recommendations on the material to be used and the construction process according to the design. Finally, the contractor executes the project according to a schedule under a reasonable budget. A typical life cycle of the project includes the three following phases: 1) design and specifications of the building infrastructure, 2) development and construction of the building, and 3) project and change management which cycles between phases 1 and 2. During each cycle, project members engage in various interactions and activities most of which is collaborative. The main characteristic of such a project include:
>
> • Individuals with diverse experience, preferences, and interest:

- AR is mostly interested in the aesthetic and comfort concepts with respect to the site. AR would recommend that "the building by the ocean should have wide windows" and "it should be of a color with small saturation; e.g., light blue".
- SE focuses on structures and materials that will form the building. For example, he recommends "the material to be concrete" and "five caisson base structures to support the building".
- CC is mostly interested in the project control aspect. They continuously evaluate budget, schedule and resources. For example, they might recommend that "a specific kind of concrete need to be used" and "testing phase should only be one month".

• Documentation and Information shared have multiple formats:

- AR sketches design concepts on drawings. In addition, he continually generates CAD plots of the site from different views.
- SE is mostly interested in graphical data on material and stress used.
- CC would like to record project schedules on tables and spreadsheets.

• Communication and Interaction Dynamics:

Being part of a team effort, peers are in continuous communication of diverse information according to their own interest and negotiation style. There are two main forms of communication (Hussein, '96) in our scenario example: asynchronous and synchronous.

- Asynchronous communication occurs when there is an exchange of technical reports and memorandum. For example, the AR might submit a memo concerning a proposal for a change in the design.
- Synchronous communication includes individual and group meetings. Such meetings are held to discuss design and recommendations to apply in the project. For example, the SE might suggest a recommendation to consider a certain amount of load on the surface.

To reach a decision on any recommendation or change, the designated team would meet and negotiate such opinions.

1.2 Problem Description

There are three leading factors for conflicts to occur within large-scale project cycles. The first one is related to the diverse nature of the participants' expertise and objectives. Professionals involved in collaborative efforts tend to bring to the table individual preferences and needs that when attempted to be translated into a specific artifact may be a source of conflict during the project development.

On the other hand, a second factor contributing to initiating conflicts is reflected in the communication aspect of information. In communicating their ideas, peers from different areas of expertise may use different terminologies that represent the same concept. This will create an overhead on the communication aspect of collaboration, since extra time needs to be spent to clear such misconceptions. In our example above, AR uses terms as 'model' or 'architecture style', while SE talks about 'structure' and 'load'. Such diversity in vocabulary was proven by Furnas (Furnas, 87) to negatively affect the human-to-human communication.

A third factor leading to conflict is related to misrepresentation of information (e.g., meeting documentation or technical reports which reflect decision making, technical procedures and design details). Most of such information is presented in single format with no consideration for the various participants preferences. It has been shown (Bodker and Gronbae, 91) that information structured according to specific preferences (data formatting, technical terminologies and design methods) would help users effectively focus on what is most relevant to their interest and would also help them communicate with their peers better.

The three factors presented define the core problem that this thesis is tackling. The next Section will describe the issues that a solution to such problem would have to take into consideration.

1.3 Towards a framework: Issues

To address the problem of conflicts in collaborative environments due to diversity and miscommunication, this thesis proposes a framework that focuses on capturing knowledge about the users, their preferences and the problems they tackle. All of this information is an essential component of a framework since traditional collaborative techniques relied mostly on the exchange of information, without references to the initiators or to the domain of interest. The proposed framework will be the basis for the development of a methodology that will give a system the capability to enhance the communication and sharing of documents between peers by adapting the presentation of relevant information to each individual preferences.

The framework identifies two main issues to be resolved: relevancy and presentation of information.

- *Relevancy of Information*

  In order to support a good communication channel across peers, a methodology implementing such framework would need to deliver the right information needed to make a decision; thus, avoiding conflicts. Here, a definition of what makes a document relevant to someone's interests need to be explored. In other words, the question would be "what are the measures that constitutes information as relevant?". The framework needs to determine a basis to find relevant information in the context of collaborative engineering projects. To do so, it needs to have knowledge about users' expertise and the problem domain. Such requirement is further explored below.

- *Presentation of Information*

  To address the problems of misrepresentation and overload of information, the framework considers two modes for the presentation of documents according to preferences. The filtering mode deals with extracting out (filtering) irrelevant information that the user may not want to see. The

second mode is to adapt the presentation of relevant information. By presentation here, it is meant to be the format and layout of data. In an engineering application, formats include the diverse ways one would show information. This can be either in paragraphs, tables, figures, graphical charts, 3D models, simulations, pictures, audio or videos clips.

Within the issue of presentation of information, the methodology based on the framework needs to have knowledge about such formats and a mechanism to allow the user express his/her preferences concerning such presentation formats.

Having identified these two main issues, the framework would need to implement several requirements towards achieving such goals. The most important requirements for the kind of knowledge that the methodology is expected to capture are both user and problem domain knowledge. Both types of knowledge are needed to resolve the issues of relevancy and presentation of information. Knowledge about the user will contain specifications on the interests and expertise which the methodology will use to find relevant information. On the other hand, the methodology will employ knowledge about the problem domain to represent specific characteristics related to the terminologies used for decision making and the artifacts being developed. Such approach will help define user-independent representations that can be used to adapt and translate information during collaboration between peers. Other requirements that revolve around the issues presented in this thesis also include considerations for modes of interactions, information sharing and collaboration support.

Based on the above requirements, this thesis describes a methodology, a model and a prototype system, called ANGELO (Adaptive iNtelligent aGEent for coLlabOratation). The system was built and modeled as a software agent assistant for adaptive retrieval and presentation of information. The World Wide Web was chosen to be the test-bed collaborative publication mechanism and repository where the agent pro-actively learns from the user about his/her preferences. ANGELO uses a statistical

machine learning paradigm to learn knowledge about the user's interests. It then uses such knowledge to adapt the presentation of information through the usage of information retrieval (IR) functionality and format layouts. Built as such, ANGELO will allow users to clearly specify their preferences and to provide a mechanism for locating documents of interest. It then adapts those documents to the user's presentation preferences (this includes layout as well as the removal of "uninteresting" information). All the tasks will be performed on the user's interface without an increased workload over his/her normal operations.

The following chapters provide more details on the proposed framework and the methodology implemented. Chapter 2 defines the problem requirements and functionality needed to be satisfied by the proposed methodology. Chapter 3 gives a background summary on related research in the fields of user modeling, software agents and information filtering. In Chapter 4, a description of the methodology and the internal workings of the model are presented. In addition, that Section gives an overview of the diverse dimensions of the machine learning paradigm used by the system. Chapter 5 describes the architecture of the ANGELO agent. Chapter 6 gives details on the implementation with a scenario of the system at work. The thesis concludes by suggesting issues to be explored in future research efforts.

1.4 Conclusion

This chapter presented a general overview of the problem being tackled here. It was shown that miscommunication and lack of standard technical terminologies caused conflicts and delays in project management. The research presented is this thesis is an attempt to address such issue.

In addition, two main issues were identified as important component of the problem domain to be considered as a solution. Relevancy of information help defines the essential issue of enabling the system find information retrieved according to user preferences and expertise. On the other hand, the presentation of information

addresses the issue of filtering and adaptation of project data and content formats to various user preferences.

2 Framework Requirements and Approach

As explained previously, the research presented in this thesis tackles the problem of reducing conflict among collaborative peers. A framework is proposed that focus on reducing problems of misrepresentation and miscommunication of information by addressing the issues of information relevancy and presentation according to preferences. This chapter discusses the necessary requirements that need to be met by the methodology and the system in order to implement a solution to theses issues.

To identify the requirements needed, the framework considers a model for collaborative interaction with focus on participants and information. Figure 2.1 illustrates a model that is composed of four entities including user and problem domain information, collaboration dynamics and information presentation. Each entity represents an essential component of the decision making process within large scale engineering projects.

The user entity includes knowledge about the participant's profile, their domain of expertise and the way he/she would want to view information and manipulate it. This module would act as the unique source for information about a particular user or team member. The problem domain component comprises of project interaction, documentation and other specific information (design rationale, or terminologies).

The collaborative component deals with the nature of communication and negotiation style used by peers during the life span of the project. Collaboration varies between problem domains, for example, the contractor might have tendency to be more authoritative and make his/her opinion most dominant during any meeting. Such meetings might take more of a chairman style interaction.

Finally, the information presentation entity is inclusive of all information generated during the course of interaction and the format that it took. It also considers the diverse modes of user interactions (manipulation or searching) with such information.

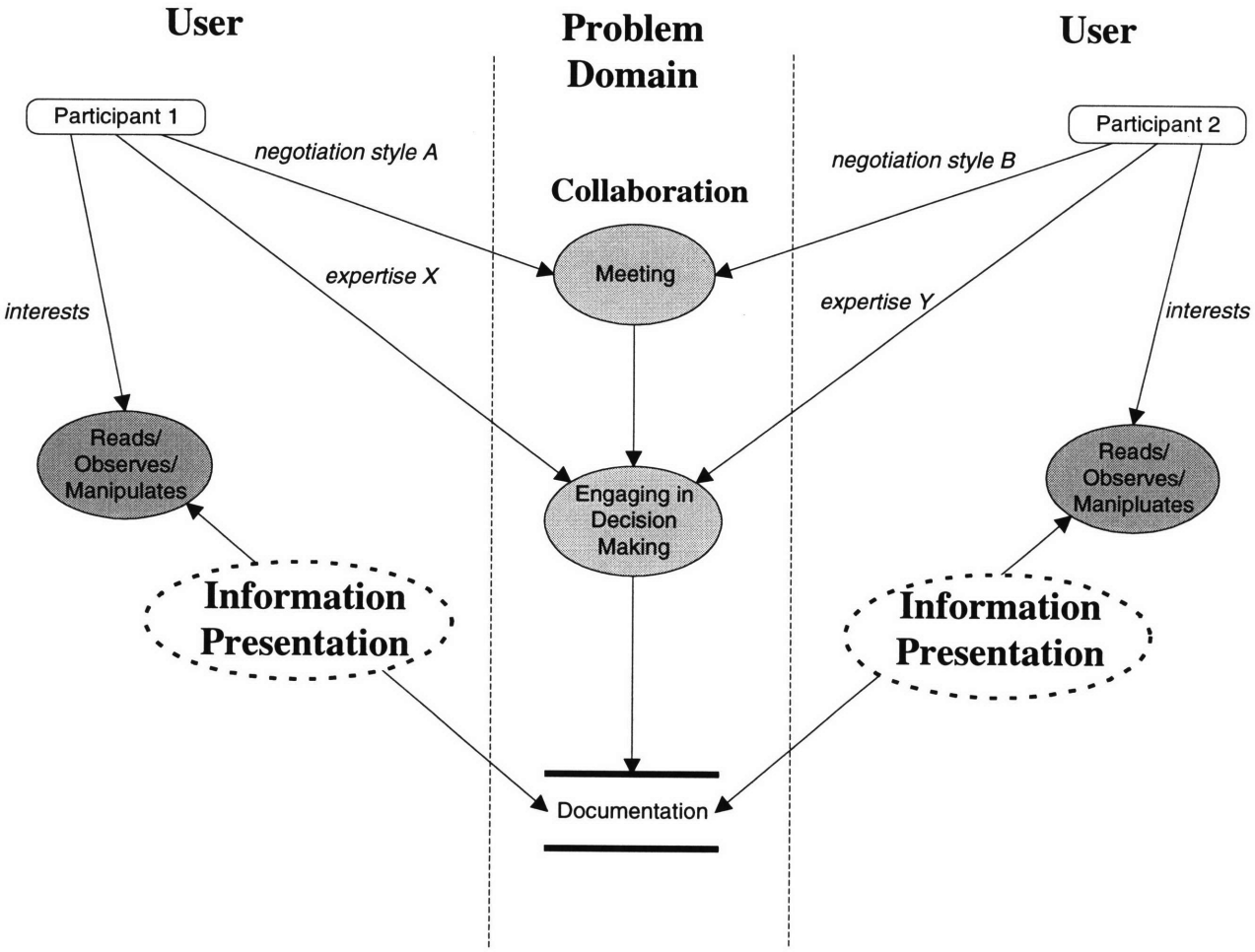

Figure 2.1. Functional Model of Collaboration with Information.

Based on the above observations from the functional model of collaboration, the next sections present requirements that constitutes the basis for the methodology implemented. Five requirements will be described which take into consideration the four entities of collaboration with information as identified at the beginning of this Chapter. Section 2.1 describes the approach needed to capture knowledge about the user preference (interests and expertise). Section 2.2 shows the effect of problem domain knowledge on improving the adaptation of information. Section 2.3 discusses how modes of presentation and interactions play a role in enhancing the capture of

knowledge about the user behavior, especially during decision making. Section 2.4 shows how the methodology could be used as a support tool for collaboration. In this case the functionality would include knowledge sharing and seamless integration within electronic collaborative interactions.

2.1 Knowledge about the User Preference

In order to resolve misinformation conflicting interactions between participants in collaborative environment, one would need to "understand" the individuals and know the characteristics that make them diverse. To achieve such goal, the methodology and the system needs to allow for the capture, storage and generalization of knowledge about the users. The system has to decide which knowledge to observe, in what form to represent it and how to manipulate it. This raises the two main issues of modeling information about the user, and learning over time new and modified information. Section 2.1.1 below describes the issues of representation and storage of information about users. Section 2.1.2 presents the requirements needed to be satisfied in order for the system to capture and learn information about the user.

2.1.1 User Modeling

User modeling refers to the capability of the methodology and the system to record a profile that captures information about the user pertaining to the context of collaboration during project development. The main features of a user model include:

- *Knowledge Representation*

  This defines the internal structure and functionality of the user model. A knowledge representation helps the designer of the system to put constraints on which user data it needs to be captured and in what format it should be stored. With good knowledge representation, one could build a system that generalizes rules about the user and therefore acts as an assistant to the user. For example, in the context of a project development in civil engineering, the knowledge to be represented is constrained to a profile of users' specific

preference in the field of expertise (whether construction management, geo-technical, or structure engineering). In this case, the methodology would represent such preference in the system in the form of concepts learned from the user.

- *Persistent profiling*

  In order to re-use a user model over time and across applications, a necessary requirement would be to have a persistent representation of the profile. This is a critical component for collaborative environments applications where the system needs to distinguish user models across diverse information. Such information can be the repository of long project cycles, which need to be shared later among participants and among other projects.

  An example of a situation where this feature is useful is in the process of human resource management during a project cycle. New comers would like to get a quick start on the project. In this case, the persistent profiling of a previous members within similar department would help the new member to be up-to-date on the issues involved in the project and the nature of the information required in each design/development phase.

2.1.2 Learning about the user

To better fine tune and automate the development of the user model, a method for machine capturing and generalizing of data about the user is required. To achieve that, one needs to take into account the following considerations:

- *Data observed and learned*

  To capture a good user profile, the system needs to learn diverse data about the user, which includes personal and login information, interests, expertise and decision making. Most of such data need to be observed while the user is interacting with the documents. The system would act as an observant of the user, capturing data needed for its learning engine as the user interact with the

system.

- *Generalization Scheme*

  For the user model to be complete, the methodology has to consider generalizing knowledge about the user. The system would create rules that will govern common user interests. It is expected that the system will use its learned rules about the user preferences to adapt the interface and customize the presentation of information. Due to the diverse paradigms in machine learning techniques, and in order to capture both statistical and rule-based knowledge about the user, the best approach to effective learning is to use a combination of various techniques. In the proposed framework, there is need to have a hybrid paradigm where a front-end supervised learning algorithm would capture user's hints/examples and a back-end unsupervised engine working as a "clearing house" by applying computational functions to cluster and generalize the data.

- *Memory model*

  Due to the dynamic structure of projects over time, one would need a form of memory to track participants' interests and involvement in different stages of the project. In order to capture preference and expertise within current interactions or project, the methodology should use a short-term memory model. In addition, a long-term memory model will need to be used for a more generalized rule set about user's decision making and preference that can be used across projects.

2.2 Knowledge about the Problem Domain

Participants collaborating in a project still do have common grounds and purpose. They meet or negotiate to design or develop a solution to a certain problem. To understand any conflict or deficiency in the collaboration, one needs to capture knowledge about the problem, its domain and its technical components. An approach

would be to put constraints on such knowledge and give the system a model representation of a project domain. This would give the agent the capability to relate user actions and decisions with the problem at hand. The sections below describe the types of problem knowledge that can be modeled and used by the methodology. Section 2.2.1 discusses the modeling of design rationale and its impact on problem definition. Section 2.2.2 describes other domain-specific knowledge that the methodology need to integrate.

2.2.1 Modeling Design Rationale

As discussed earlier (Section 2, Figure 2.1), the problem domain level of collaborative large-scale engineering projects involves mostly a interrelated set of decision making components. Figure 2.2 below shows the DRIM model (Peña-Mora, 95) which offers a good modeling of design rationales and recommendations.

DRIM main concepts include:

- *Designer*

  This data structure represents the entity engaged in the decision making. Such entity in the problem domain proposed is a project participant which can be an engineer, such as structural engineer or it can be an architect.

- *Proposal*

  This refers to the recommendation or information suggested by the participant. The *proposal* is a rich representation which includes alternative opinions, component description (what the proposal consists of), reactions to other proposals and versioning.

- *Intent*

  Such information refers to what the designer's goal or purpose of the recommendation made. An intent maybe either an objective to be reached (e.g., to protect concrete from damage), a constraints to apply on an artifact (e.g., specific standard measurements for ensuring compliance of a bridge), a function to show a behavior or a goal to be reached by designers (e.g., to

construct support for a bridge).

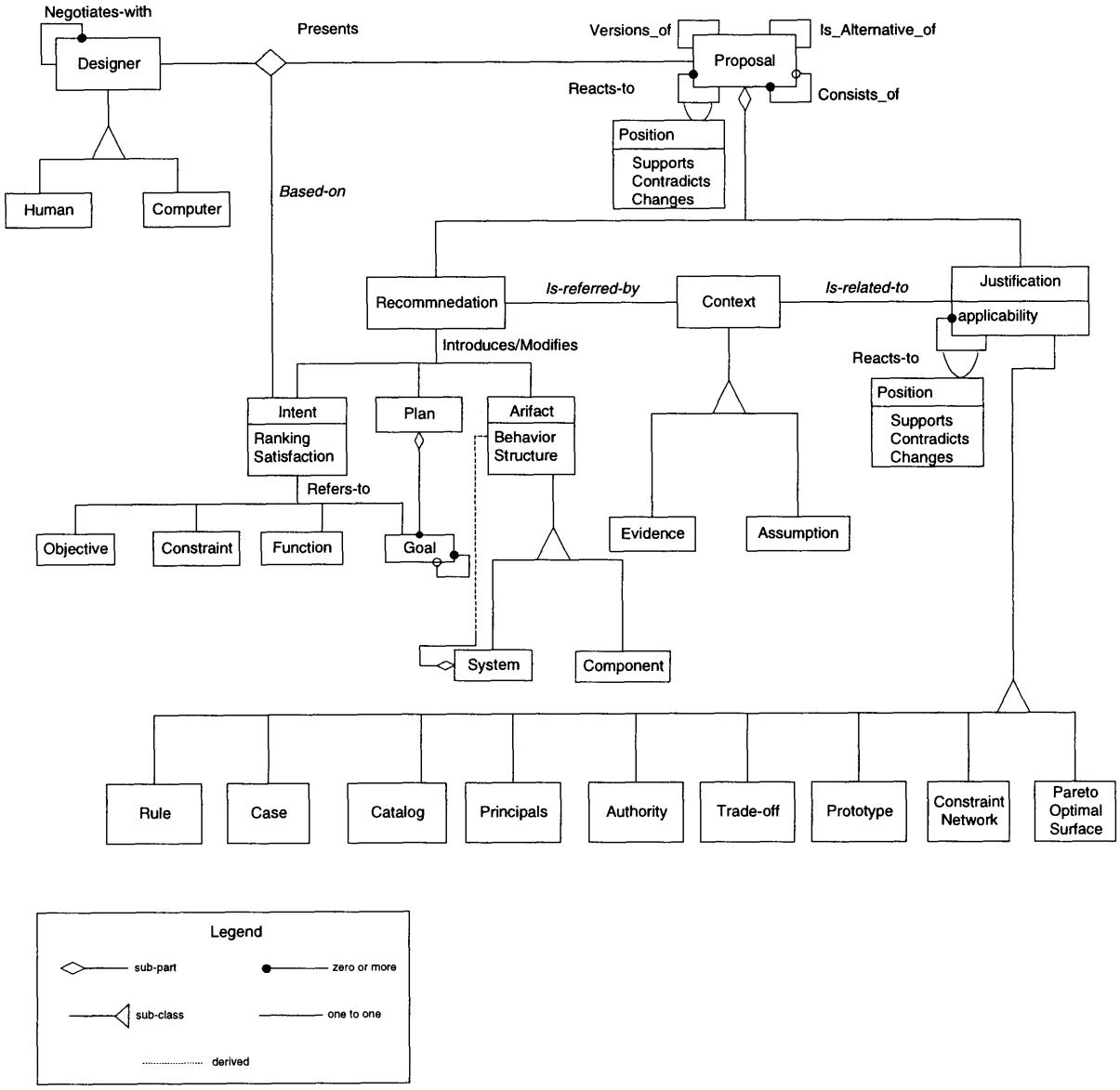

Figure 2.2 Functional Model of Collaboration with Information.

- *Recommendation*

  This represents the result of selecting an artifact, applying modifications on artifacts or for the suggestion of new plans.

- *Justification*

  A justification is the reason why a certain recommendation will satisfy a recommendation. This would refer to the initiative taken by the source (engineer or designer) to take a decision (recommendation). An example of such reasons of initiatives include either an authority level, a design rule, or a certain standard.

- *Context*

  This represents the information generated during the design process. DRIM models that in two forms. One being as an evidence that is based on some facts, while the other called assumption is based on data quality and its presentation.

It is suggested for the DRIM methodology to be used in coordination with the ANGELO methodology to provide design rational and intent information. Such information will help improve the system widen the scope of learned user preferences to include recommendations and intent.

2.2.2 Other domain knowledge

Looking at a large-scale project as a problem with specific domain, one would need specific information about the terminologies and concepts expected to be used in that domain. For example, in a bridge construction project, one would expect the system to recognize terms like "slab" or "pier". In addition, the methodology need to have knowledge of high-level constructs from such concepts. This information will include the physical objects dealt with in the problem. For example, there has be a definition of a "bridge" as "a formation of a slab supported by two or more pier".

In order for the system to adapt the presentation of problem information to the individual use, it should also have knowledge about those data formats and how to translate them. For example, if a construction manager have preference to look at bridge information as tabulated data that includes material and cost, the system should translate a graphical information and extracts the data relevant and then present them

in another format. The knowledge needed in this case would be a rule to map from one artifact information that contains geometric and material data to another tabulated format. On the other hand, the object representing the budget may have different views that could be called depending on the user preferences. These views may be one of a graphical nature while another can be of a tabulated form.

2.3 Modes of Information Presentation and Interaction

2.3.1 Modes of Interaction

Another dimension to this requirement is the need to consider not only interest, but also modes of operations and interactions that the user might be engaged in. In each case, the system would need to adjust the relevance according to the context and the level of interest expressed by the user.

- *Skimming Mode*

  In this mode, users might express interest in certain topics that may not have the same importance since this will be more of exploring and searching mode. The system will consider every document read as related and rate such document according to interest expressed.

- *Meeting Mode*

  During a collaborative meeting, the user is either receiving recommended information or suggesting it to others. The system would not rate any document received (recommended by others) as relevant until the engineer rate it as so.

- *Authoring Mode*

  In this case, the system would mark the authored information as relevant since the user had some level of authority and interest.

2.3.2 Other presentation observations

Based on the two main issues of relevancy and presentation of information (as described in Chapter 1), there are two modes of information presentation. The first is basic filtering of information, where the methodology extracts irrelevant documents or entities within documents. Secondly, in addition to looking at information content, format is an essential element for adaptation. The system should be able to apply the rules learnt from the user model and the problem domain knowledge to adapt the documents in a format that is of interest to the user.

In addition, the system could consider patterns in user's interactions with other peers and/or with documentation. For example, the system can detect time spent on a page, number of private messages to certain peer, or user interface actions (e.g. printing, cut/paste). Such features would help the agent enhance its learning over time and generalize better.

2.4 Collaboration Support and Seamless Integration

Since the nature of the problem presented in large-scale civil engineering projects involves collaboration among peers in order to reach decisions or to exchange recommendations, the methodology and system presented should be applied within a computer-supported collaborative environments. The requirements that need to be considered include the ability to share knowledge across, and integrate the adaptation of presentation with the communication channel. Both such issues are detailed below.

2.4.1 Knowledge Sharing Among Peers

Individual expertise and recommendations produce the knowledge that users would like to share across with other peers. In addition, newcomers to the project would need to get a quick status and information on the project cycle and the individuals involved. The system should be able to use the persistency of user model and allow different clients to coordinate and communicate the shared data. Such requirement calls for the need of an inter-system collaborative protocol.

2.4.2 Support Tool for Electronic Collaboration

The methodology developed will act as a user-centered support tool, aiding users in adaptive retrieval of their own relevant information. In order to provide the collaborative component, the system should be integrated with current electronic systems. Such integration should be seamless in a sense it will not distract the available communication paradigm. The methodology presented here would need to integrate with such system to offer users the capability to present recommendations of relevant documents. For example, during a design brainstorm session, one engineer would suggest a recommendation and back it up by facts from previous documentation. The engineer would send the document to an architect and the project manager who in turn would get an adapted view according to their own perspective on such recommendations. To be more specific, a construction engineer might present a graphical plot of stress load on some structure. In this case, the architect will get a filtered document (removing the information on stress) but still the project manager would get an adapted tabular form of the project resources indicating a certain problem.

2.5 Conclusion

This chapter discussed the necessary requirements that need to be met by the methodology and the system in order to implement a solution for the two main issues presented earlier in Section 1.3. Such requirements included capturing user and problem knowledge, observing modes of information presentation and supporting collaboration and knowledge sharing. The requirements analysis was based on a model of collaboration among engineers.

To better understand the area of research that this thesis belongs to, the next chapter gives an overview on the general area of software agents and on related research to the requirements presented above.

3 Survey of Related Research

Having presented issues and requirements to address the problem, this Chapter describes current research in the areas related to the functional requirements described in chapter 2.

3.1 Learning Interface Agents

3.1.1 What's an Agent?

An agent can be considered as a computational system that has long-lived states with goals that acts from sensors and effectors. It is autonomous, therefore giving it the capability to decide which actions to take in its current situation to maximize progress and outcome (Russell, 95).

3.1.2 Software Agents

Software agents are particular type of agents, inhabiting computers and networks, assisting users with computer-based tasks. Most of these agents co-exist as a front end to software applications in which users need help either automating repetitive tasks or filtering information (Maes, 94).

The basic characteristics that distinguishes software agents from other programs are the following:

- *Pro-active*

  Agents have the capability to take initiative on some actions. For example, an e-mail agent can delete unwanted e-mail on the behalf of the user.

- *Long-lived and autonomous*

  The autonomous behavior is achieved by the fact that software agents should work independently from the user most of the time, and should decide which next step to take. User input is only considered as examples for the learning process but not instructions on which action to take next. An agent is long-

lived since it is designed in a way that it can sustain itself over a long period of time.

- *Adaptive*

  Agents are encoded with learning algorithms in order to adapt to a certain environment or in order to influence its autonomous behavior to do the right thing (Maes, 94). Machine learning theory is used to reach the adaptive behavior, which generally can mean that the agent while observing its environment it is learning new concepts and generalizing rules of behavior.

- *Personalized and customized*

  The adaptive behavior can also have a more broader sense, it can also mean that the agent is adapting to the user, and therefore making itself personalized to a specific user.

3.1.3 Kinds of Software Agents

Software agents can be categorized in three dimensions. Figure 3.1 shows such classification. The X scale represents the measure of user interaction that the software requires. On one hand, the low level of interaction means that the program is designed to assist the user and automates a lot of task. On the other hand, the high level of interaction means either the user is programming an application or the software has a graphical interface that needs a lot of input from the user (e.g. direct-manipulation visual programming or navigating in a hypermedia space).

The Y axis reflects how much agent technology (as defined in Section 2.1.2) is used in the application and the scale of the involvement in automating software tasks. Example of these tasks might consist of a database query, or a graphical interface to a CAD system. The more the agent is involved in the software, the more it automates software tasks and learns about the user, and therefore achieves more adaptation.

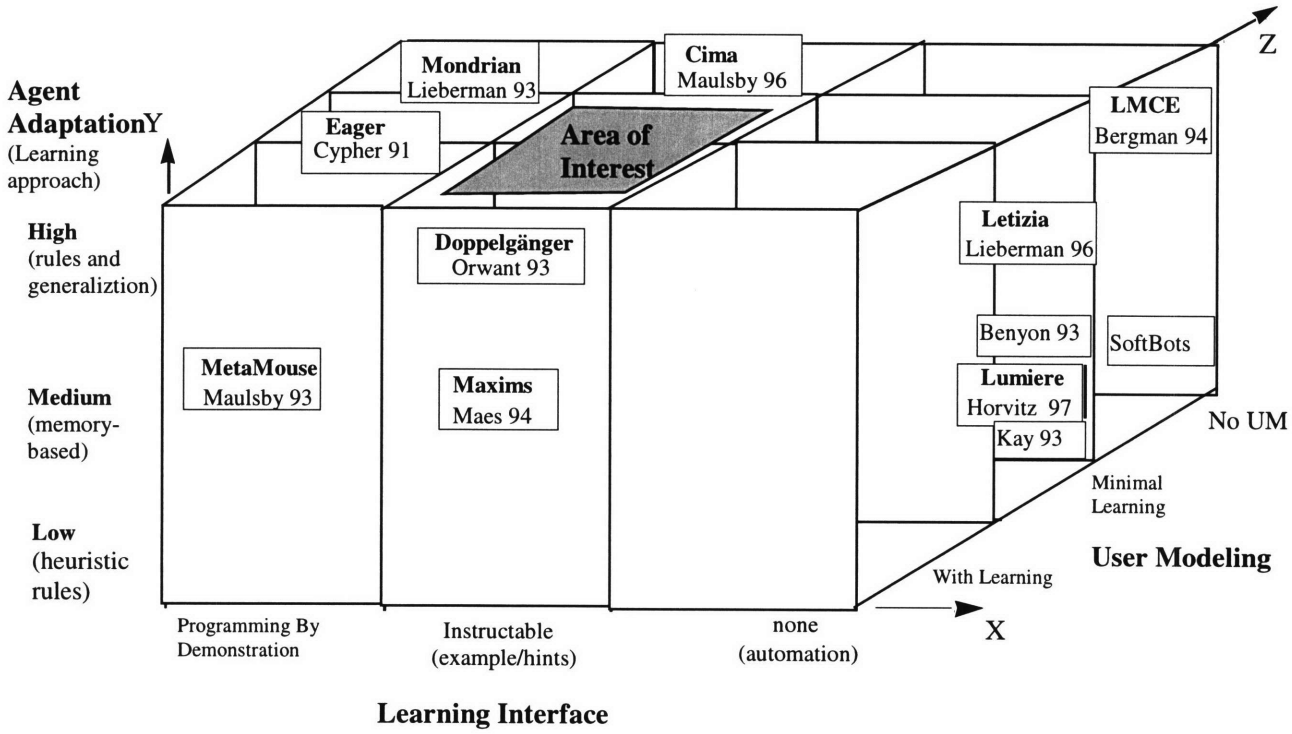

Figure 3.1. The Software Agent Research.

The Z axis shows the level of agent to agent communication or exchange of information (between two or more agents). Moving from the lowest level on this axis to the highest, different categories of agents with more distributed capability and agent-to-agent communication are found. For example, agent on the Web are highly communicative in nature, while an interface agent running local on a desktop is on the low scale. In between, are the agents who have some local and network based capabilities. The most common agents in this category are the social filtering agents like HOMR (Maes, 95) and FireFly (Maes, 96).

3.1.4 Learning in Software Agents

Learning in a software agent varies and depends on the application used, level of user involvement and complexity of tasks. For most interface agents, there are at least three categories of techniques or approaches used: programming by demonstration , memory-based and machine learning. Based on the categorization in Figure 3.1, a

description of each approach will be given along with examples of the systems using such techniques.

- **Programming By Demonstration Agents**

Figure 3.1 includes agents that get instructed by the user through a programming by the demonstration approach. They use ad-hoc learning techniques (such as simple look-up tables with a matching algorithm) to create simple rules governing the user's actions. In addition, they use hard-coded knowledge about the domain, but lack the capability of adapting to the user. Examples of such systems are Cypher's Eager system (Cypher, 91), Lieberman's Mondrian (Lieberman, 93) and Metamouse (Maulsby, 93).

- **Example-Based Learning Agents**

Such agents memorize situations in their environment (user interface) and compute the predictions accurately after a good amount of training. For example, Maxims, an e-mail agent for Eudora (Maes, 94), uses Memory-Based Reasoning (Stanfill and Waltz, 86) to capture all the user's e-mail actions. These user patterns are stored in a table that can be used by the agent for predictions of future actions. Maxims does not generalize or build rules about the user. It just records observed data as items in the look-up table. In order to adapt well in the longer run, Maxims needs a lot of training. This affects its performance. It uses a lot of memory and operates slowly.

- **Knowledge-based Agents**

Some interface agents are built using classical behavior-based machine learning algorithms like genetic programming (Koster 94), or rule-matching techniques (Winston 93). Other agents use rule-based learning. Most of these systems in this category use symbolic-based reasoning that generates rules to be evaluated through predicates. They tend to do better adaptation and generalization than the techniques mentioned earlier. Example of such systems is Cima (Maulsby, 96) and Bergman's (Bergman 94) manifest causal agent which uses a combination of computational learning (Schema mechanism) and rule-based representation.

- **Information Filtering and Retrieval**

IF and IR are two techquines used for document browsing. There are several similarities and differences (Belkin and Croft, 92) between the two. Although they both process documents in similar matter (indexing), one applies filtering on a dynamic stream of information and the other relies more on a static database (IR). The amount of data involved in IF is mostly large and unstructured, while with IR it is structured and pre-indexed most of the time.

Information retrieval have three main paradigms: (i) statistical, (ii) semantic and (iii) contextual. The first approach emphasize statistical correlations of word counts in documents and document collections (Salton, 83). Semantic indexing is another example of statistical approach but used to capture the term associations in documents, so offering a closer meaning to the text. The last paradigm is a boolean logic one, where one encodes structural relationships among terms (Gauch, 89). A simple example of commercial IR systems are the search engines that exist on the World Wide Web.

Information filtering (IF) can help the user by eliminating the irrelevant information and by bringing the relevant to the user's attention. Filters are mediators between the sources of information and their end-users. Users can build their own filters from rules. Social filtering is a recent approach (Liang, 96) where IF selects documents based on the ratings that other users assign to them.

- **Towards better Personalized Information Agents**

Personalized Information Agents (PIA) refers to software agents that aim to assist the user find the right information that suits his/her interest. Such agents employ information filtering techniques and adaptive behavior. FireFly (http://www.firefly.com) is a social filtering information agent but lack the adaptive behavior and needs lot of instruction to become personalized to a particular user (see Figure 3.1). NetAngels (www.netangels.com) employs a good neural net-based IF

algorithm, where users can train the agent to find the right information. It lacks the personalization aspect and the pro-activeness characteristic.

Another kind of PIA is Letizia Web-agent (Lieberman, 95) which uses pure observation-based techniques, where it monitors the user's web browsing patterns and indexes those entries for a "look-ahead" search. It does not use any machine learning at this stage. Compared to the suggested requirements, Letizia lacks the user profiling capability and the agent-agent communication. However, it does offer a good feedback, whereby a window shows the agent a view of Web pages being browsed to assist the user.

The Newt system (Sheth & Maes, 93) is a truly PIA and it uses genetic algorithms to dynamically adapt the agent to the user most readable documents. It sill lacked a user instructed feedback and does not adapt the actual presentation of the information. The focus is more to have a good user profile.

The research proposed in this thesis is the development of a software agent that is similar to a personalized information agent but offers more adaptation and better user-centered information filtering and retrieval. It captures a user profile, and maps its adaptation to a new presentation of the interface with the closest information matching user's interest and in the desired form. The sections below give a more detailed description of the model, the algorithm and an early implementation of the system.

3.2 Work on Knowledge Modeling and Sharing

Chapter 2 presented issues and requirements to address the problem of conflict and user preferences, this Chapter describes current research in the areas related to the functional requirements described in Chapter 2. The research described in this thesis draws on the intersection among three areas of research: knowledge collaboration, user and problem modeling and interaction context. Figure 3.2 below shows a three dimensional chart where each axis represents a grouping criterion and the research related to it. The first grouping (i.e., X-axis) covers research in knowledge

collaboration and sharing. The second grouping (i.e., Y-axis) is focused more on the work related to knowledge modeling research, which includes user and problem modeling. Finally, the last grouping (i.e., Z-axis) is on the interaction context, when systems take into consideration diverse mode of user operation and the nature of the adapted presentation. Below, each subsection describes the different approaches of related research efforts and evaluates them according to the requirements presented earlier in Chapter 2. The survey also emphasis the offerings of the ANGELO methodology in comparison with the other related research.

3.2.1 Work on Knowledge Modeling (Users and Domains)

This category includes systems that acquire knowledge about the user or the problem domain of concern. Such acquisition can have an element of learning in it or it could be pre-defined or pre-coded knowledge in the system.

In term of user modeling, two major issues research are whether the systems use learning or just simple knowledge. Earlier attempts on building user models did not use machine learning, but focused on building a good knowledge base about the application as well as the user and the problem domain (Kay, 93). In this case, user interaction styles were mostly predefined in the system (Shneiderman, 95). For example, in intelligent tutoring systems, the user model focused on a student model. Thus, it is difficult to use that same model for another type of user such as an instructor or even use it in other systems that may not have the same heavy bend on tutoring. Another example of a system with pre-coded knowledge is the Lumiere project[1] (Horvitz, 97). Hard coded heuristic rules were used for the agent, greatly limiting the ability of the agent to memorize or generalize. The system was built using a one-year worth of knowledge collected by 30 people observing users individual interactions with Microsoft Office applications. Thus, the agent relies mainly on a "unified" user's model built using Baysian network as the description for the

[1] The Lumiere project was integrated in Microsoft ®'s Office 97™ product as the "Office Assistant".

knowledge. The agent does well in prediction the user's goal, but lacks any learning about specific users.

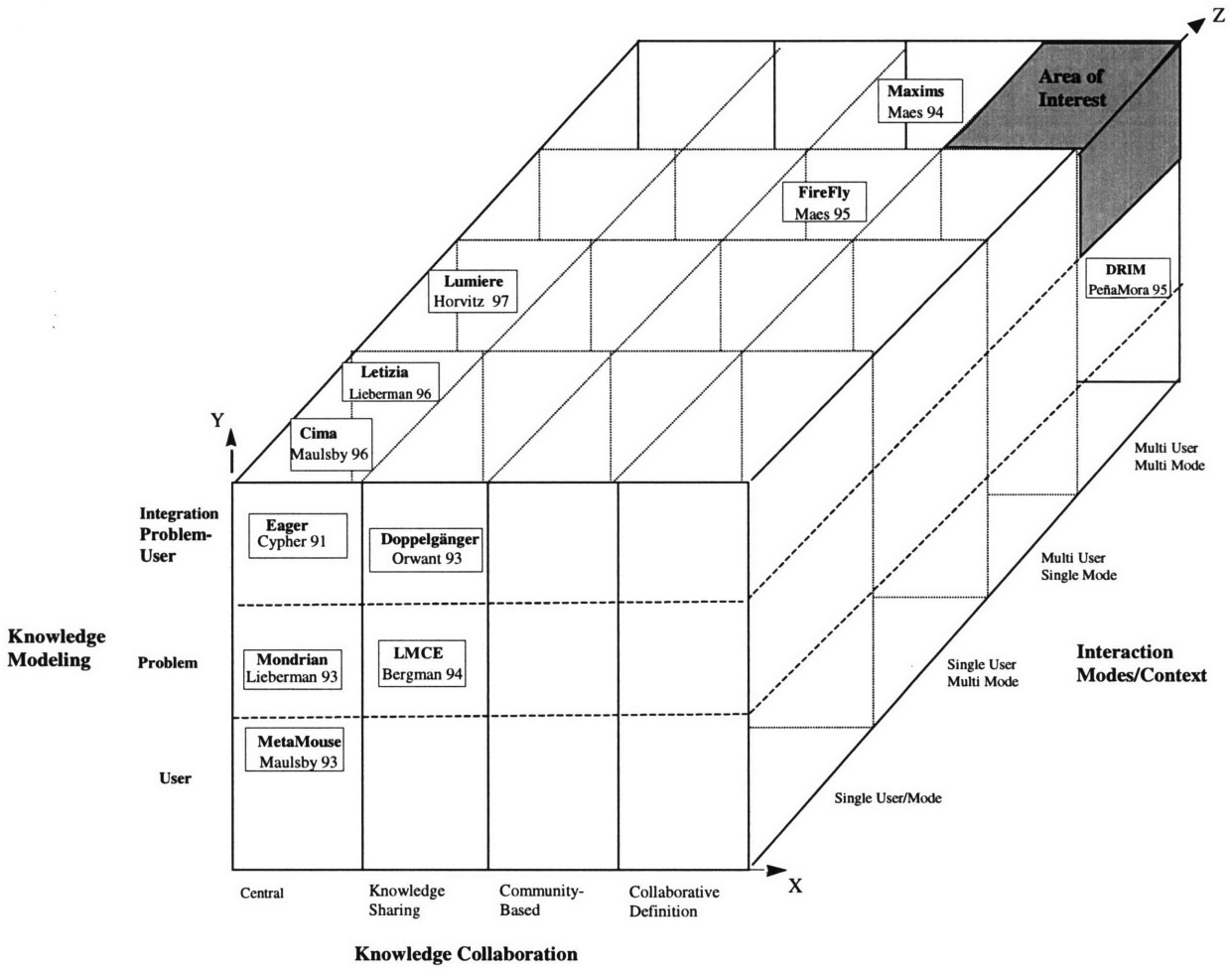

Figure 3.2. Research in Knowledge Modeling,Collaboration and Interaction.

Other systems avoided the "hard-coding" of knowledge and used machine learning to model the user in order to allow for the individual record of user interactions. Some of these systems include Maxims (Maes, 94), Eager (Cyhper, 91) and MetaMouse (Mauslby, 93). Maxims is an e-mail agent for the Eudora mail system, it uses Memory-Based Reasoning (Stanfill and Waltz, 86) to capture all the user's e-mail

actions. These user patterns are stored in a table that can be used by the agent for predictions of future actions. Maxims combines user/problem domain since it has pre-defined rules about e-mail domain (sort, read, delete e-mail) and it observes user action. It also have a bulletin-board option where users can share their agent's knowledge to improve the learning of their assistants. However, Maxims does not generalize or build rules about the user. It just records observed data as items in the look-up table. In order to adapt well in the longer run, Maxims needs a lot of training.

Doppelgänger (Orwant, 93) also combines problem domain and user modeling. It uses a simple "like/dislike" interface where the agent stores user data and clusters them according to a utility function. The design lacked a good design to allow such information to be shared among users. Another system that combined problem domain and user modeling is Cima (Maulsby, 96). Cima is considered a more advanced system since it uses a rule-based approach to learn from user's hints to the agent. In this case, the agent builds rules for generalization while observing hints from the user. The hints observed are considered as instances of basic features, which combined, will form rules. With that implementation of Cima, the level of problem domain associated with the user model was minimal. The agent learnt low level tasks (like move object on screen or edit text) that were not mapped to a higher level problem driven representation (like interest in bridges, or recommend a rationale).

Bergman's LMCE (Bergman, 94) was an agent that had little knowledge about the user or the domain but was programmed to explore its environment (i.e., Macintosh Windowing system), and learn in an unsupervised and generalized way. The system generated rules based on Action/Selection pairs, trying to learn an action from a certain world selection. Generalization was possible by combining several rules into a higher level one. However, such system lacked any user input, so it took long cycles to learn a small concept as "how to iconize a window". It also didn't have an option for it to tackle hints or examples from the user that would help it perform better.

3.2.2 Knowledge Collaboration

Being in a collaborative environment with repository data, a user/problem-centered system would need at times to communicate with other agents. This is done in order to exchange recommendations on documents, or to cooperate on certain tasks. Such interchange of information is implemented in some systems that employ inter-agent communication. It was also shown (Maes 95, Liang 96) that such kind of information exchange would improve the learning and adaptation performance of the agent.

There has been several approaches to implementing distributed communication among agents to gain more learning. We describe two main approaches below: social filtering, and collaborative meeting environments.

An example of such social filtering approach is FireFly (Maes, 95), which automates the word-of-mouth process, and learns about the user's taste or opinion in the world of entertainment. It uses that information to best serve the user's needs. FireFly uses nearest neighbor calculation in order to suggest new music that might interest the user. The system finds a close match to someone else's taste in music and recommends the items. Thus, FireFly is applicable and feasible in collaborative recommendation-based interaction, where the system would build a pattern of users' opinion clustered together. Still, FireFly lacks the capability to capture knowledge about the relevancy and presentation of the data recommended, since it solely relies on statistical recommendation of other users, with no integration with knowledge about the problem.

Another way for systems to share information is through a collaborative process of defining such information (Peña-Mora, 96). This takes another twist, in that the systems negotiate the knowledge and exchange artifacts. This can be done either by human users or computer-assisted agents. CAIRO and M-RAM (Peña-Mora et. al., 96) uses such paradigm.

3.2.3 Interaction Modes

There has been little work on considering different modes of operation that users might engage in. A mode of operation is defined as the cognitive and perceptive state that the user is in while engaging in a collaborative design process. For example, Maxims (Maes, 94) looks at user's actions and is able to detect different reading styles since the user's interactions patterns are captured and clustered in memory. Another example, the Lumiere agent (Horvitz, 97) has knowledge about a large amount of possibilities that a user might be doing. It uses this information to give to the user-filtered information and the most relevant one.

3.3 Conclusion

Having examined the related research in the area of software agents for information retrieval and collaboration, one can draw some conclusive remarks regarding the deficiency of such systems. It is notable that the area of applying problem knowledge with a user profile have not been explored. In addition, the issues of presentation and modes of user interaction are not totally investigate by most related research.

4 A Framework For A Software Agent To Assist in the Adaptation of Information Retrieval

As proposed, the requirements and research issues, presented in the previous Chapter, point to the problem of capturing, memorizing and presenting user-based preference- influenced documentation. The methodology developed in this thesis is an attempt to address such a problem. The implementation is an interface agent that learns from the user and adapts its document retrieval to specific preferences and styles. This Chapter explores and describes the architecture and the components that constitute the system.

4.1 Architecture

The proposed architecture is based on the requirements discussed in Chapter 2, and it is shown in Figure 4.1. The system consists of three main sub-modules: Learning Engine, User Model, and Information Filtering algorithm. Each module provides a functionality towards achieving one of the agent's goals of assisting the user. A more detailed description of each module is provided below.

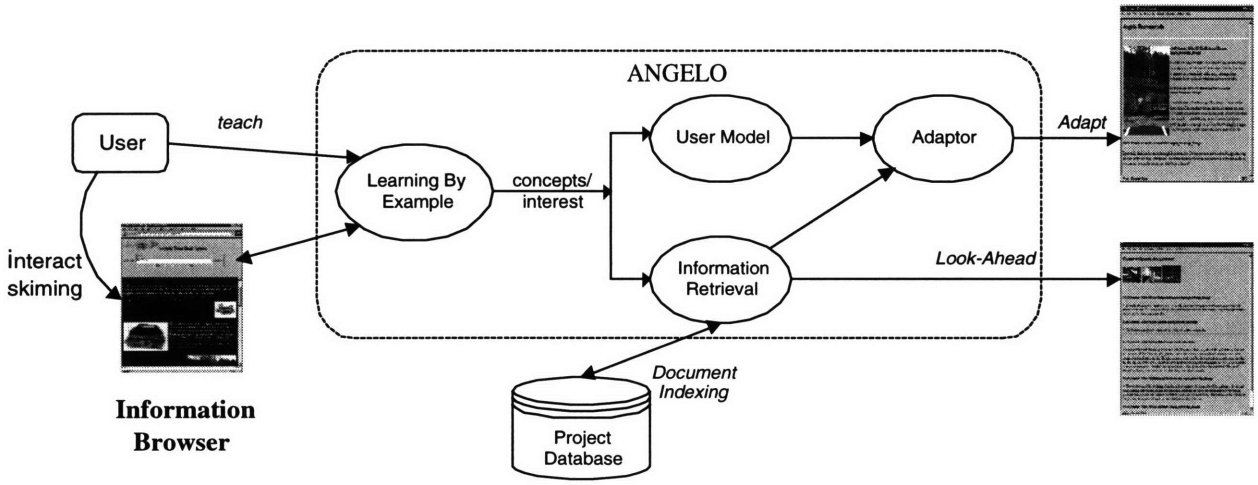

Figure 4.1. Architecture of ANGELO.

The client side of the system consists of two components – one for capturing user preferences and another for displaying the user-model, filtered page, and further suggestions. The server side consists primarily of the tagged documents as well as a savant engine (Rhodes & Sterner, 96) server that performs the relevance calculations on text information. The server also maintains files that contain the user profile. The Information filtering/retrieval engine operates on a separate server, which indexes and tags documents. Thus, the actual site data and the persistent user model reside on another server.

4.2 Learning Engine

Based on the necessary requirements laid out in Chapter 2, ANGELO was designed an instructable agent that is user-centered. It learns by observing user's actions, and taking hints/concepts from the user. ANGELO also uses clustering techniques to assemble the rules/concept pairs that will be stored in the user model. This Chapter describes the learning engine.

4.2.1 Instructing the Agent while browsing

In order for ANGELO to capture knowledge and update the user model, the user needs to instruct the agent on his/her interests. In addition, the system should be able to communicate with the user and report its status. For example, it should tell its user how much it has learned so far, and why it is taking these decisions. ANGELO employs a "learning by example" methodology (Maulsby 96) that gives the user an active involvement in teaching the agent new concepts and the user interest associated with those concepts. This technique will be defined here as "teaching while browsing" (TWB), since the interface is seamlessly integrated with the content of information. TWB's functionality includes:

- *User teaches the agent some semantic concepts behind HTML objects and expresses his/her interest associated with those concepts.*
- *The TWB interface is integrated seamless within the browser, therefore allowing the user to teach the agent most document objects while browsing.*

- *TWB records concepts from the user and the associated action. It then generates a feature vector for later processing by the clustering engine (see Section 4.1.2).*

The image in Figure 4.2 below shows the user interface of the TWB functionality that ANGELO employs. The system added extra HTML tags in order to provide a minor interface that the user can use to choose a particular object to teach the agent about. For a text element, the user can click on a red button attached to the paragraph. As for an image, the user can click on the image itself.

The actual Angelo interface consists of a text field for the user to enter his/her own concept definition of that object. However, ANGELO provides a predefined concept that has been obtained by the Savant engine from the caption of the object or the text surrounding the object. It also has an "interest" scale (Low to High on a scale of 5) from which the user can choose. The algorithm used in ANGELO to apply the TWB methodology is presented in Figure 4.3 below:

4.2.2 Clustering Algorithm

The learning algorithm for the ANGELO agent is based on a simple statistical clustering technique using C4.5 (Quinlan, '93). The methodology defines an HTML document as a set of features on two levels: both page-wide features and local object-based features. Page features takes into account content and layout styles that distinguish an HTML-based document from another. For example, a main page for a Web site would have more images than the actual content. Such observations are mapped into coded rules. This is shown in Figure 4.4 below.

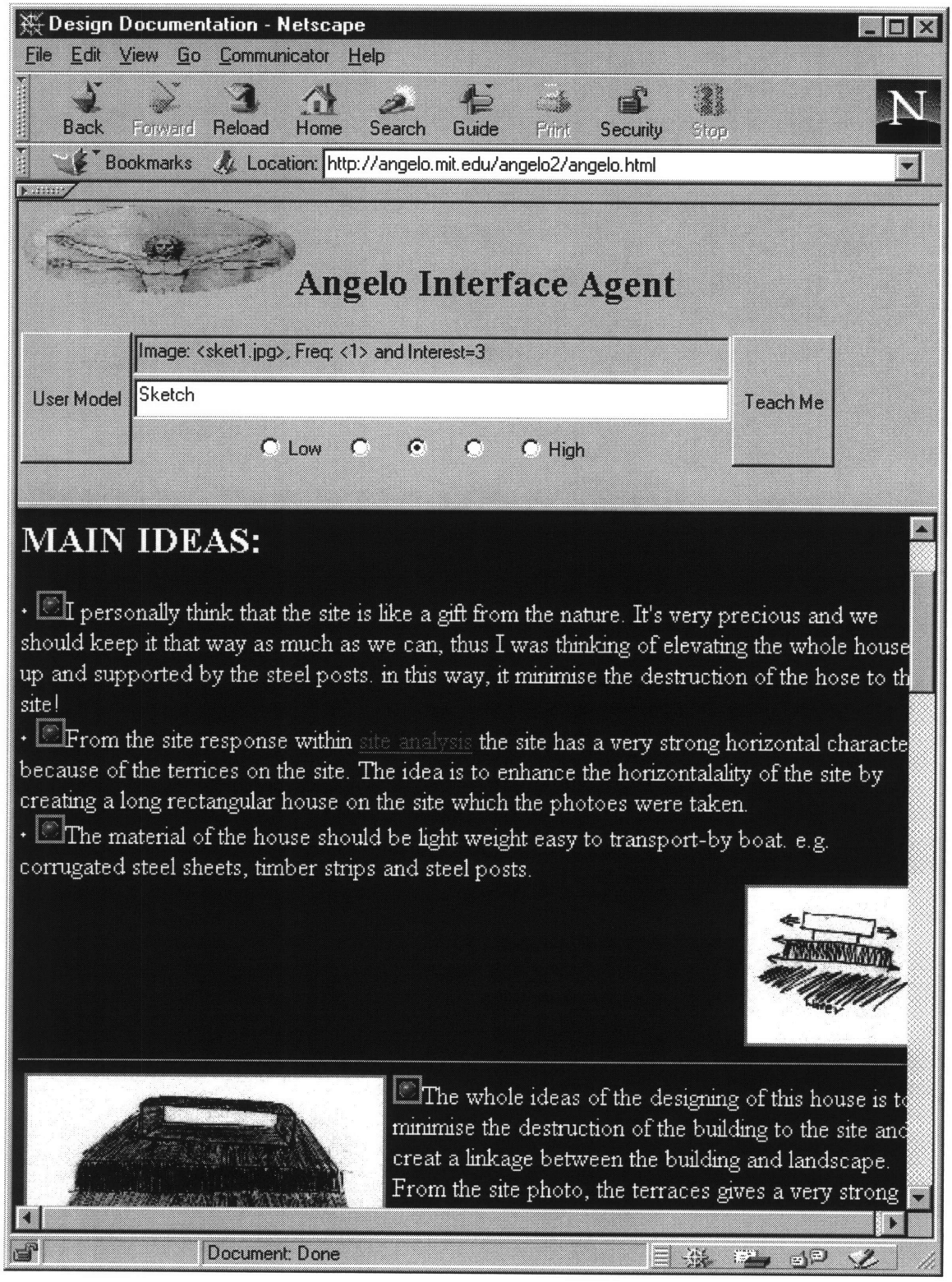

Figure 4.2. "Teaching While Browsing" Interface.

Action: User clicks on an HTML object (image or bullet next to text)

If Text element:

Angelo queries the Learning Engine for previous occurrence of that object's concept and interest.

If no previous occurrence then
The Savant algorithm returns a keyword index of the paragraph.

Update UI

If User enters a new concept or expresses interest then Angelo updates the savant engine memory.

If Image element:

Angelo queries the User Model for previous occurrence of that object's concept and interest.

If User enters a new concept and interest
Angelo creates a new record in the UM history.

Angelo adjusts the interest.

Figure 4.3. Basic ANGELO TBW algorithm.

4.3 User Modeling

In an effort to personalize the skimming process, the system uses a user model to dynamically store the learned interests and behavior about the user. ANGELO's User Model (UM) includes a history-like record of the concepts that the user expressed interest to while skimming. For example, the user might choose an image and instruct the agent that this is a picture of a house. In this case, the system captures the concept house, and puts it as an interest to the user.

1) Page Features:

- *Title*
- *Type*: (layout description – currently only two rules)
  Rule1: If there is a map then Main Site Page
  Rule2: If there are many links then Site Index Page
- *Concept*: Word vector generated by Savant engine as well as a neighboring documents.

2) Object Features:

- *Name*
- *Type*: IMG, TXT, URL, MAP.
- *Attributes*: (Length/Size),Width, Height, Aspect Ratio
- *Concept Vector*: Hand coded for Images, Savant generated for paragraph.

Figure 4.4. Feature Vector for the learning engine in ANGELO.

While continuously observing the user, ANGELO updates the User Model with the concepts and the interest frequency that the user chose. In order to provide a feedback on the user model, the user has the option of displaying a visual window with the concepts that the agent is learning (see Figure 4.5 above). The concepts display each instance of data that the user categorized as such concept and sorted in order of preference. Such a visual representation acts like an agent feedback that is helpful for the users to realize the status of the agent's learning.

In reference to short term versus long term memory issue (discussed in Section 2.2), the user model in ANGELO's current methodology implements the idea of short term, since it stores most recent user interactions and interest. On the other hand, clustered information (using C4.5 as described above in Section 4.2.2) that the user commit to is stored in a second stage as a long term memory.

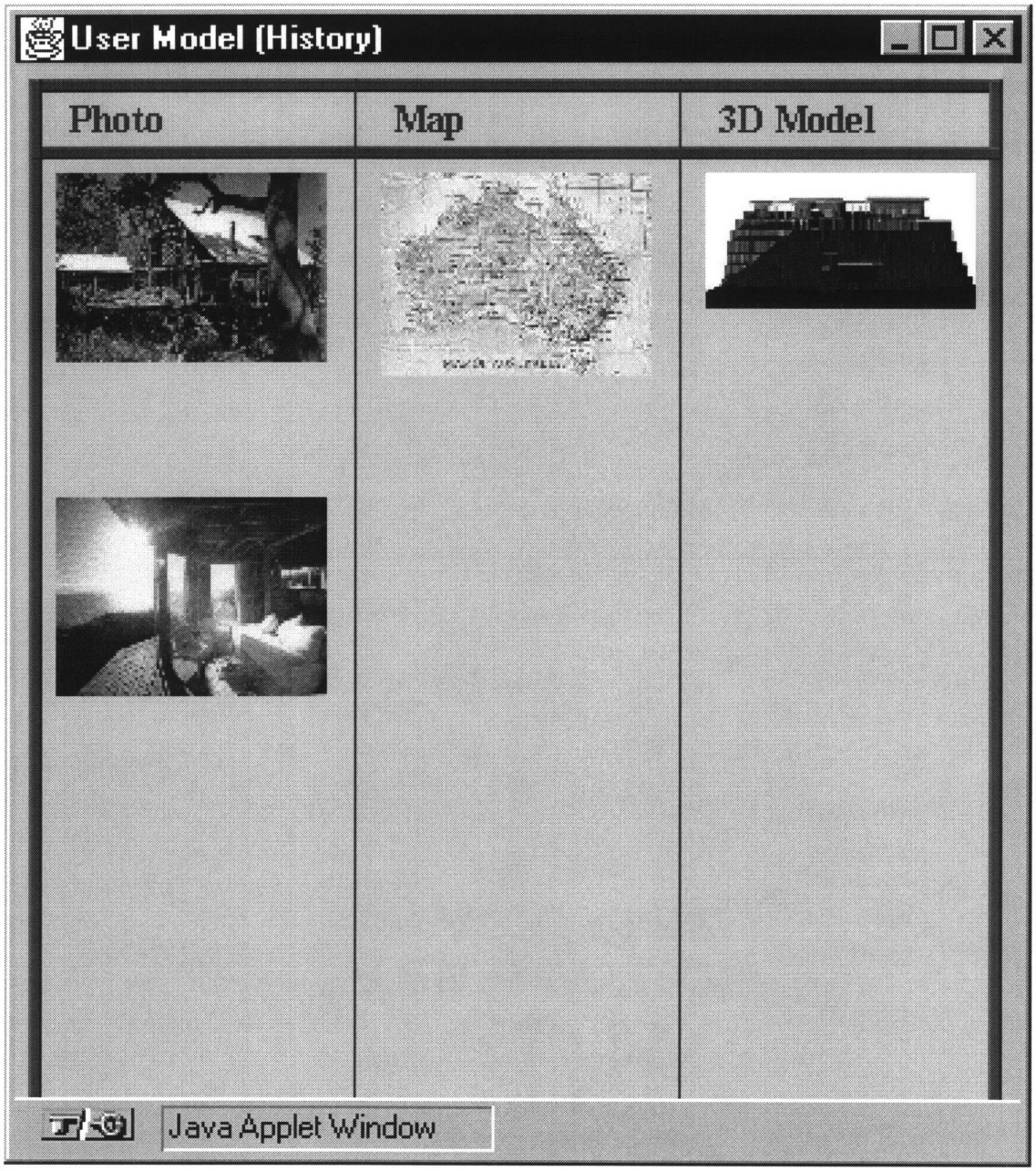

Figure 4.5. User Model Display Window.

4.3.1 Representation in the User Model

The internal representation of the user model is a classified list of user concepts coupled with document keywords and level of interest. The list is a set of vectors (V_i), where each vector is composed of a list of user concepts (UC) correlated with a

document keyword list (W) and an interest (*i*) which is a scalar from -2 to 2. The scalar is an indication of how much the user is interested in a document entity. A scalar of 2 indicates high interest, a zero means no opinion and –2 means don't show ever.

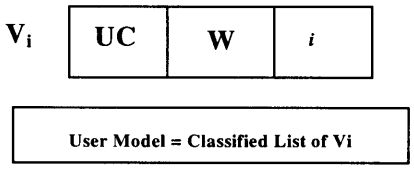

Figure 4.6. Representation in the User Model.

4.4 Information Filtering/Retrieval (Savant)

The Savant engine was adopted from the Remembrance Agent (Rhodes and Sterner, 96) to generate a vector of key words from textual information. Such vector is used to generate the set of concepts on paragraphs, figures, captions and page levels. These features are stored in component structures. These structures are then loaded into the classifier database and clustered. After clustering, C4.5 (Quinlan 93) provides a certainty measure that is also recorded in the component structure. Each component also has a value of the interest the user has in the component.

The interest values (weighted by the certainty measure) of all the components in a given cluster are summed and the result is placed in the interest field of the cluster object. New components can then be easily classified into the clusters and their user interest level can be determined for the filtering and suggestion mechanisms.

4.5 Modes of Operation

Based on the methodology described above, ANGELO has three main modes of operation. The Savant engine feeds ANGELO with the relevant results from both a

filtering and a retrieval perspective. This gives ANGELO the capability to have both Presentation and Look-Ahead modes.

- ***User observation***

  The agent will act as an observer in the beginning to collect as much data needed about the user. This observation is carried out in the background, where the agent captures most of the interface events, keystrokes, and concepts learned from the TWB interface. The observed data is stored by the agent and passed to other modules.

- ***Presentation and Adaptation***

  Having a good knowledge about the user, the agent will be able to adapt the presentation of documents to fit the user's interests. This is accomplished by filtering a document according to rules generated from the user model. This is the agent's capability of generating human-readable text output (such as HTML) from its memory.

  The presentation mode is the filtering out unwanted concepts or data. The Adapter module serves to assemble a decision based on the learned concepts in the user model, and the output of relevant documents from the Savant engine. It then generates a recommendation document that contains only the relevant objects that the user might be interested in.

- ***Look-Ahead recommendations***

  Knowing the user's interests and preferences, the agent can suggest other relevant documents that the user might have seen in the past or that might exist in the hyperspace of related documents. The Look-Ahead window serves as a recommendation to what other documents that exists in the same repository have related concepts of interest. This is done by applying a query to the Savant engine on all the documents.

4.6 ANGELO's Object-Oriented Model

Based on the previous analysis given in Chapter 2 of the requirements of knowledge capturing and presentation of user and problem knowledge, and the description of the methodology given earlier (see Sections 4.1 to 4.5), an object oriented model for the ANGELO system was designed. The model encapsulates information about a user and the problem concepts as well as interaction with the system and its operation. The model also shows the relations and associations that user and problem knowledge models can have (e.g., ANGELO/DRIM interchange) and between shareable knowledge (short and long term memory). Below is a description of the ANGELO model illustrated in Figure 4.7.

4.6.1 User Modeling

The *User* class acts as the user model which includes personal information (name), system information (id, password, last time logged on, time spent on system) and the concepts that the user expressed interest in. The *User* class gets updated by *Angelo* module while the agent is observing the user.

4.6.2 User/Problem integration

4.6.2.1 Problem domain as Concepts

A Concept is a class encapsulating high-level information learnt from the problem domain. The name of a concept acts like a semantic label or tag for the set of HTML objects which belongs to that category of a specific concept learned. In addition, the Concept class includes information concerning the user interaction and interest to that concept. Frequency reflects the number of interactions and "*interest_level*" is a measure set by the user. In ANGELO, a set of Concept instance objects gets initiated by an automatic generation from the Savant engine, through the keywords extraction algorithm. As will be explained further later in Section 4.6.4, the Concept class updates the memory module with the appropriate information satisfying the correct conditions of short/long term memory models.

4.6.2.2 User/Concept Interaction

Chapter 2 showed the importance and novelty of this thesis approach in combining both user and problem knowledge in order for the ANGELO system to learn and adapt the information according to both user interests and problem domain specification. The model designed is based on such observation. As Figure 4.7 shows, The *Concept* classes is manipulated by the User directly and indirectly. The former is when the user expresses a specific concept for a certain interaction. For example, the user might choose to rename a previously assigned concept "site" to a certain map image and call it "map location". The indirect or explicit interaction is when ANGELO presents a concept for a text or image, and the user would confirm the validity of such association (see "Validates" relationship in Figure 4.7).

4.6.2.3 ANGELO/DRIM Integration

Another feature that ANGELO was designed to accomplish is the use of a Design Rational and Intent Model (see Section 2.2) as the problem domain model of collaborative engineering. Figure 4.7 shows a portion of the DRIM model (colored in gray, and placed on the top of that figure) that maps or relates to some classes in the ANGELO model. Such an approach is useful to establish an integrated solution for collaborative design and negotiation by including a system to adapt presentation of documentation.

4.6.3 Sharing Knowledge

For ANGELO to be a support tool for distributed collaborative environments, the model needs to offer an extension to a multiple user model and shared memory space. figure 4.7, the model includes a "Shared_With" attribute for an association between one user to another.

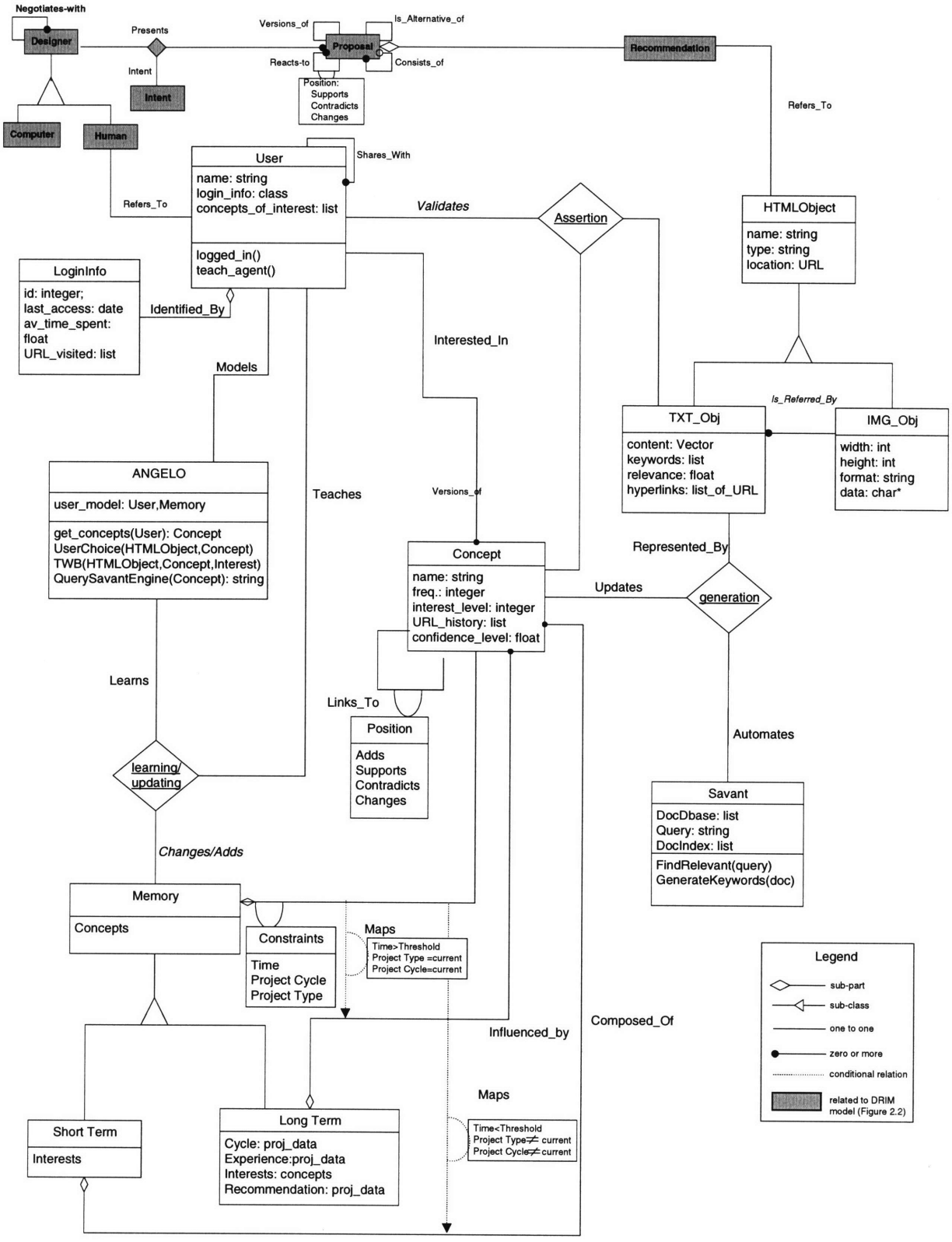

Figure 4.7. A Model of ANGELO

4.6.4 Memory Model

Memory Model is one essential time component of the ANGELO model. It encapsulates the short tem and long term memory of a user or a group project. Each short/long term memory will get updated by *Angelo* according the constraints of time, project cycle and project type. For example, when the project is in a transition from design to implementation and the team changing, ANGELO should get an update message either from one or multiple users. In that case, ANGELO would transfer the conclusive, and generalized recommendations to the long term memory. The short tem memory is continuously updated with the current state of user/system interactions and currently observed concepts.

4.6.5 HTMLObject Class

The HTMLObject class refers to the set of basic representational entities that the system need to capture and model as a concept instance. There are two types that ANGELO supports ; text and image objects. Each are subclasses of the HTMLObject, and gets instantiated when the browser loads the HTML document referring to those objects.

4.6.6 ANGELO Class

The ANGELO class acts like the kernel of the system, holding all instances of the above classes. In addition, the learning and retrieval methods are implemented in that class.

4.7 Conclusion

This chapter gave a description of ANGELO's design elements that form an object-oriented methodology. The architecture and the components that constitute the system were explored. Such architecture included several modules that captures user information and adapts the content by filtering out non-relevant information. The object-oriented model described in section 4.6 showed ANGELO's design as a set of interconnected classes encapsulate information about a user, problem concepts as well

as interaction with the system and its operation. The model also showed the relations and associations that the user class and a problem knowledge models (like DRIM) can have and between shareable knowledge (short and long term memory).

5 ANGELO's dimensions of Machine Learning

This Chapter explores various dimensions of the machine learning methodology presented in the previous Chapter. According to (Grecu & Brown, 96), machine learning in design have seven basic elements that need to be explored. Such dimensions offers a framework for the employment of a machine learning in an agent-based collaborative design environments such as the one envisioned to be used in conjunctions with ANGELO. The following sub-sections describe the dimensions of learning in the system presented and offer further criticism and extension for such framework.

5.1 What can trigger learning?

The problem being addressed here (see Chapter 2) focuses on improving conflict during the design process, by capturing users' preferences and expertise so that conflicts due to misrepresentation of data could be minimized. The ANGELO system is designed to continuously observe user input while interacting with design documents. In other words, the user has indirect control over the learning process, and learning is triggered by user interactions with a document presentation system such as a browser. Thus, the more the user interacts with the system (which include document skimming, object/concept selection and communication with other users), the better the system learns specific user preferences and extends the user model.

5.2 What are the elements supporting learning?

After defining the situations that trigger learning, one would need to define the problem domain elements and forms that the learning engine would have to analyze for possible learning (Grecu & Brown, 96). Chapter 2 presented the user and problem domain requirements which the system has to take into consider in its learning process. Such knowledge include user preferences (concepts of interest, graphical

representations, and expertise) and problem specific information (for example, design rationale and recommendation information).

5.3 What might be learned?

The previous sub-Section identified domain elements that is supported by the learning process. Still to be defined are the entities that need to be learned by the engine. Such entities come from different levels of the problem domain. From a high level (meta-level) information about the design, designers and communication aspect (e.g., negotiation, and meetings) to a lower level like design and project data (e.g., charts, schedule, and maps).

In ANGELO the learning engine focuses on resolving the specific problem of conflict among peers by capturing and learning user information (preferences, recommendations and negotiation style). Such information is of a high-level form (based on concepts and knowledge representation of the problem at hand, see Chapter 4). In addition, ANGELO's learning engine learns low level data about the design documentation at hand which includes document layouts, design data and content (see Section 4.4).

5.4 Availability of knowledge for learning

Since design involves communication and collaboration components among peers, it is important for user-centered agents (such as ANGELO) to share their knowledge learnt about the problem domain at hand. Availability of such knowledge is determined by the amount of interaction as user has with the system. However , more knowledge can be obtained by using knowledge from other users interacting with the system. It was shown by (Bergman, 96) that agents can achieve better learning by sharing their knowledge in those cases. Defining how and in what form such agents will communicate and/or transfer the learnt data becomes of great importance.

In order for ANGELO to act as a collaborative support environment (see Section 2.4), it shares its knowledge in diverse manners, which includes:

- *Direct transfer of knowledge between the agent and the user model.*
- *Indirect communication between two ANGELO agents with the assistance of an inter-mediate CAIRO (Hussein, 96) agent, within a design session..*
- *Through persistent storage of user model, agents can share such model with other agent using a query mechanism..*

5.5 Methods of learning

There are various methods of learning that agent-based systems can use. The specific choice depends on the problem requirements, elements supporting the learning and the type of data to be learnt (Grecu & Brown 96). In the domain of support tool for collaborative design process, in one hand, the learning method uses knowledge-base learning approach for high level concepts like design rationale and on the other hand it employs numerical and statistical methods for low level data manipulation.

As was shown in Section 2.2, ANGELO uses a model that combines both high-level and low-level learning methodology. It is actually designed around the idea of long and short memory. Long term memory intended for a more generalized and persistent mode of learning. It uses C4.5 algorithm that extracts features from the user/problem interaction to generalize on the user model. Short term memory is meant to include the most recent concepts learnt. The current method for this purpose is based "learning by example" paradigm and works a front-end to the long-term memory learning engine (see Section 4.2).

5.6 Local vs. Global Learning

Learning in an agent-based system can occur in a centralized or distributed collective fashion. In other words, an agent can learn its own set of data and share that with other agent who compete to adapt and learn better. A decentralized learning is

where a collective outcome of learnt small data from each agent forms a global learnt model of the problem. ANGELO uses only a local learning approach for each user and a distributed learning within a agent society where each has a specific role as a support tool for the collaborative design process. The DaVinci paradigm (Peña-Mora et al, '96) is formed of various agents allowing for distributed learning; one for the communication and meeting coordination (CAIRO), another for the multi-reasoning on design cases (M-RAM) and for user preference and profiling (ANGELO).

5.7 Consequences of learning

Learning and adaptation of an agent improves on the design process by playing the role of a support tool. Therefore, it would assist, automate and recommend during the design process. Specifically, ANGELO assists with the management and presentation of information by learning about user's preferences and exercise. It can also act on the behalf of the user during redundant design meetings, once it formed a generalized and significant memory and user profile.

5.8 Other dimensions of ML in Design

The described dimensions based on (Grecu & Brown, 96) do not cover all aspect of applying machine learning in collaborative design. The following are some points and dimensions that this research has identified as very important and absent from the original framework:

- *The Agent should have the capability to evaluate and measure its performance according to the problem solution required.*
- *The dimensions proposed does not take into consideration change of design, data and peers across time. Such radical and rapid changes will have negative impact on the learning if it was not dealt with before.*
- *The dimensions also don't propose how agents can communicate and what type of information they can share. ANGELO is designed in mind to offer a level of protocols among agents in order to give the users the option to choose which information they want their agent to share with other agents.*

- *There is the question of how the learning method can deal with better generalization and expansion. There is a narrowing effect that happens when the agent is solely learning from user interaction over similar document without spanned explorations.*
- *The framework of Grecu and Brown does not take into consideration design cycles dynamics and how it would affect learning. This thesis proposes the long/short term memory model with different learning methods in order to handle recent un-clustered user information learned (short term) versus high level, problem related knowledge (long term).*

5.9 Conclusion

In order to place the use of machine learning (ML) in ANGLEO within the area of Machine Learning in Design, this chapter explored various dimensions of the machine learning methodologies which ANGELO uses.

This chapter endorsed the MLinD model (Grecu & Brown, 96) for the basis to evaluate the user of ML in ANGELO. The chapter also pointed at several deficiencies in that model which did not capture all the ANGELO's features.

6 Implementation

Based on the previous observations and requirements for the problem, ANGELO was built as a software agent that co-exists within a Web browser. The Internet was chosen as the medium for collaboration and distributed information. ANGELO is implemented as a Java Applet coupled with a set of JavaScript routines that interact with the Netscape browser. The major difficulties in the design of the system were the integration and communication between the browser and the agent. In this Chapter, we present details on the implementation of the system, and describe a scenario demonstrating ANGELO's capability.

6.1 A Java-based Agent Implementation

ANGELO is implemented as a set of Java classes with a main applet that interfaces within the Netscape browser. It uses Netscape ONE technology to communicate with the Web browser.

6.1.1 AngeloApplet class

The applet called *AngeloApplet* is loaded by an HTML start-up file. The applet initializes the learning interface and adds a button to launch the User Model component. AngeloApplet class uses the Netscape Internet Foundation Classes to call upon Javascript function that would open two windows (recommendation and lookahead window). Figure 6.1 shows the three windows open. The window on the upper left contains the main AngeloApplet interface running within an HTML frame in the browser.

When the user clicks on the User Model button, AngeloApplet launches a separate thread for the UserModel class, which in turn get displayed as a Frame. On the other hand, the user will choose the "Teach Me" when he/she wants to express interest in a certain HTMLObject in the lower browser window which is displaying the site content.

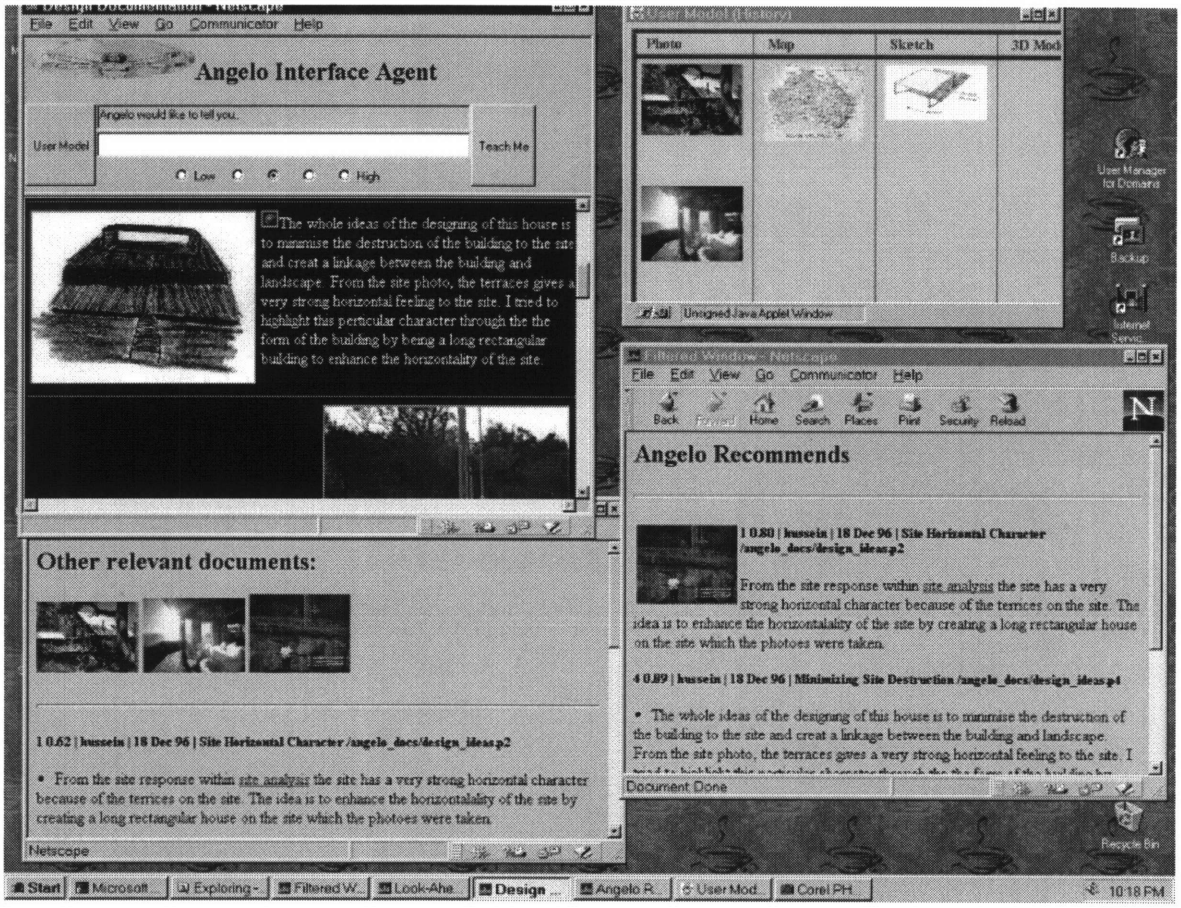

Figure 6.1. The ANGELO agent integrated with a Web browser.

6.1.2 UserModel class

The UserModel class play the role of a memory profile of the user's interest and as a visual display of such memory. While the user is teaching the agent, the UserModel updates its window with the set of concepts learned and the User's interest about those concepts. Internally the UserModel class also stores choices and frequencies about those concepts for later use by the agent to apply C4.5 algorithm or any other heuristic measures in order to adapt the interface and retrieve relevant information.

6.1.3 Engine class

The Engine class is the interface to the Savant information retrieval program as described in Section 4.4. The latter is C-program that runs as a cgi executable on the

same HTTP server that launched the AngeloApplet applet. The Engine class sends a query string to the Savant program in the form of a URL. Upon receiving the query result in the form of a list of keywords and documents found, the Engine class would parse this stream and extract relevant documents and their frequency. It would then update the look-ahead window accordingly.

6.2 System Description

The main window shows the user input frame as well as the main browsing frame. The user-model (see Figure 6.2) shows images that have been selected in the past and their relative rating by category. The filtering window (see Figure 6.1) shows the current document with text that is not considered interesting by ANGELO removed (the text and images retained are accompanied with a user weighting.). Finally the suggestion window (see Figure 6.6), provides a summary of interesting document segments from those provided by the server that not on the document requested by the user but on other documents on the same server. Those are also rated according to relevance to the current user interests.

6.3 Scenario

This section describes a scenario using ANGELO for a collaborative project on the construction of a building in Sydney, Australia. Data were presented through a Web site (S.Wu, '95), where users can browse and look at the design process so far. The scenario presented here is of a geo-technical engineer who just started getting involved with the project. The engineer has expertise in identifying problems with site location, and was given the task to judge on the location and give some opinion on the initial design of the architect. One might expect that the user's interests will be in the site location and documents about the design. Thus, media of interest to him/her would be maps and photos.

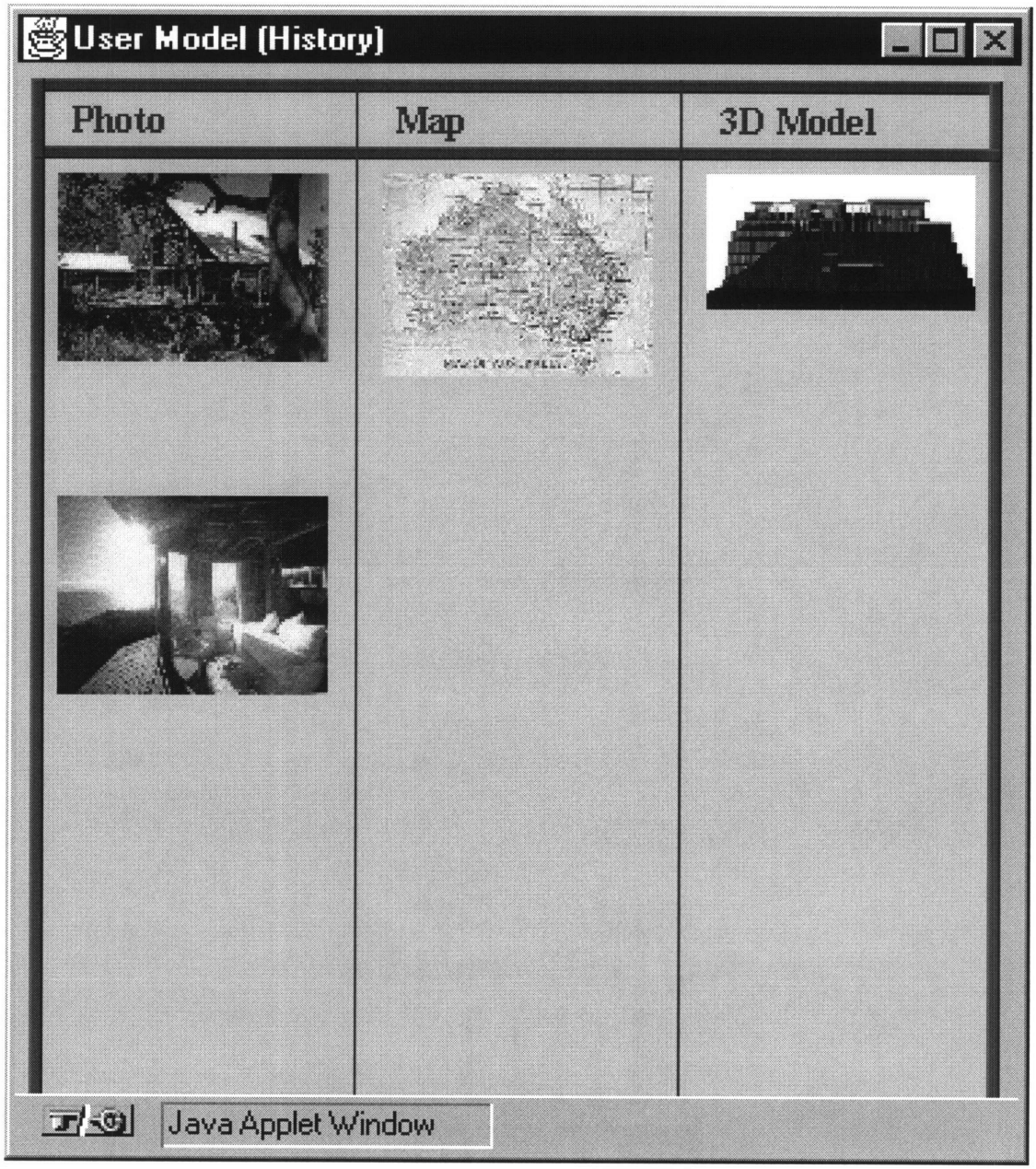

Figure 6.2. User Model Display Window.

Initially, the user would browse the project data, starting from the first document called "Site Analysis & Site Model." User expresses interest in the second paragraph describing the site, and a map showing the geographic location (see Figure 6.3 below).

To express his/her interest, the user clicks on site photo and gives the agent a rating of "High". After clicking on the "Teach Me" button, the User Model (UM) learns a new concept "Photo" and gets an icon representation of such concept (see Figure 6.5 below).

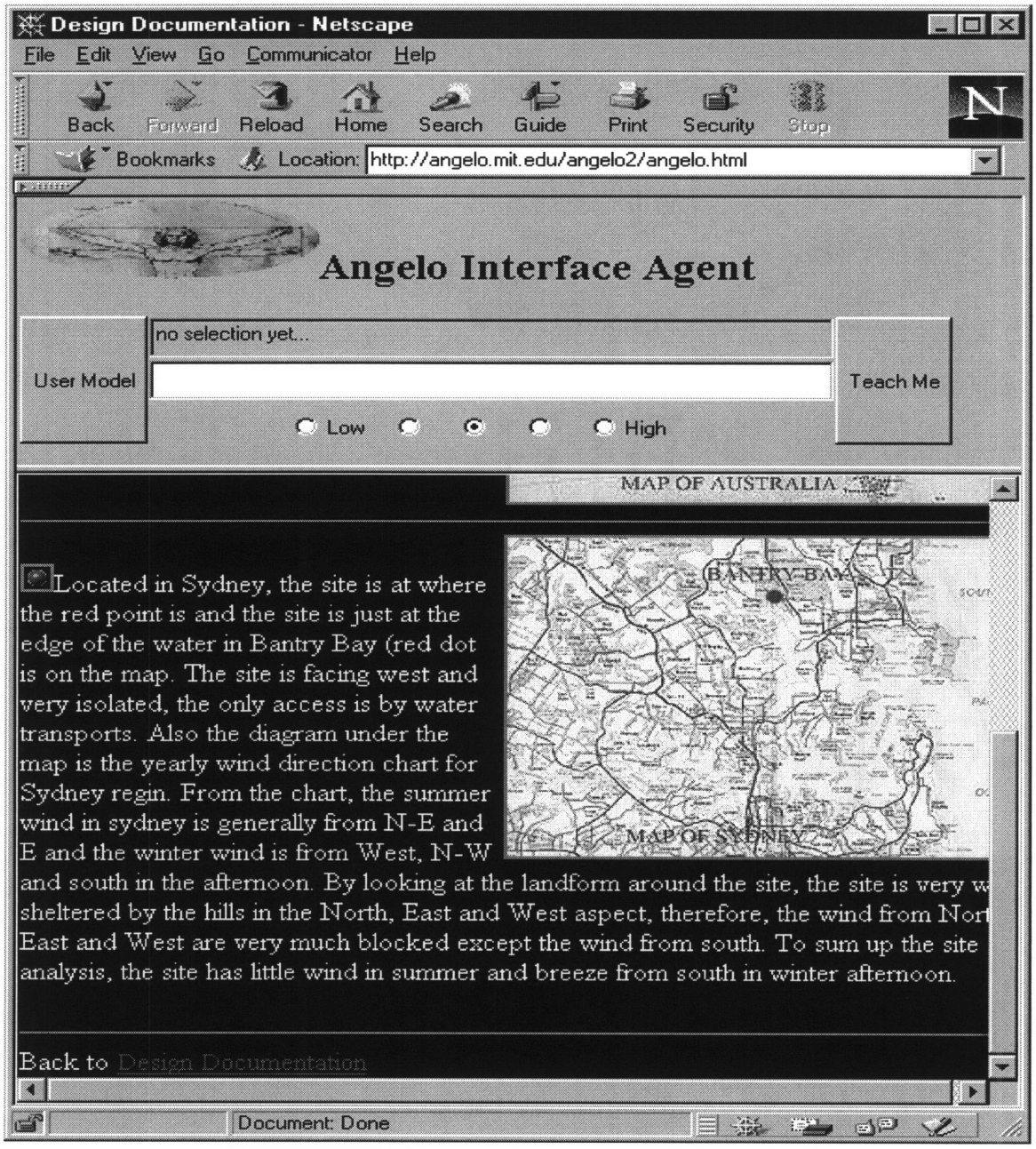

Figure 6.3. Text info on Site location and map picture of the lcoation.

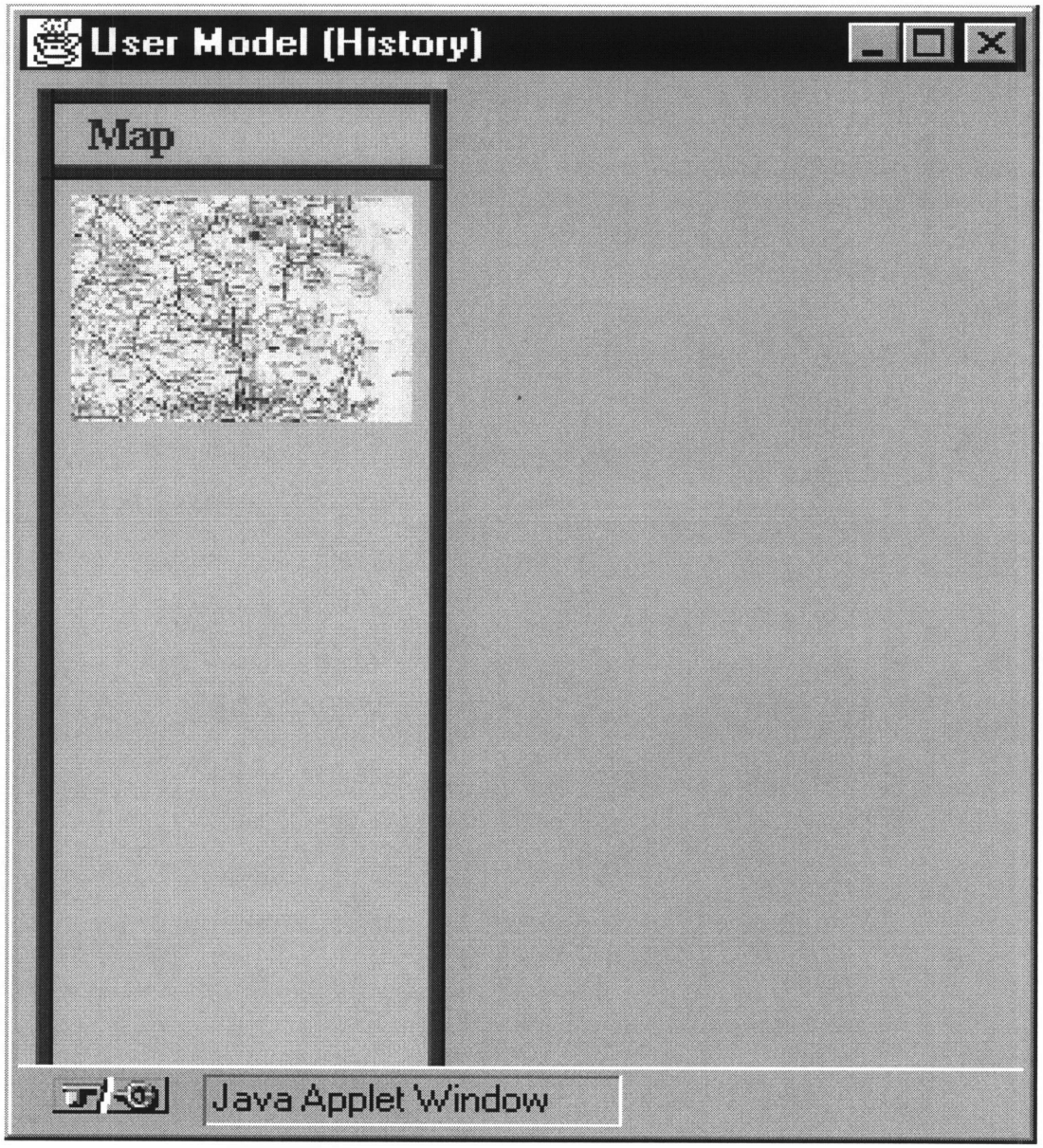

Figure 6.4. User expresses interest in Map concept which gets initiated in the user model.

As a result of the new addition in the UM, the agent will update the adaptation window by filtering out irrelevant information to the concept Photo, and therefore

showing only pictures of maps and text related to the site both from the original document and other documents on the same server (see Figure 6.5).

To recommend other relevant objects that the user might have interest in, the Engine module calls upon the Savant program with a query that includes the most recent concept user interacted or expressed interest in. Upon getting a result back from Savant, the Engine would update the look-ahead window with actual documents, and images that appeared to have close match to what the user might be interested in. In this case, when the user choose to look at site photo information, ANGELO would display (as shown in Figure 6.6) all relevant images and documents existing within the repository.

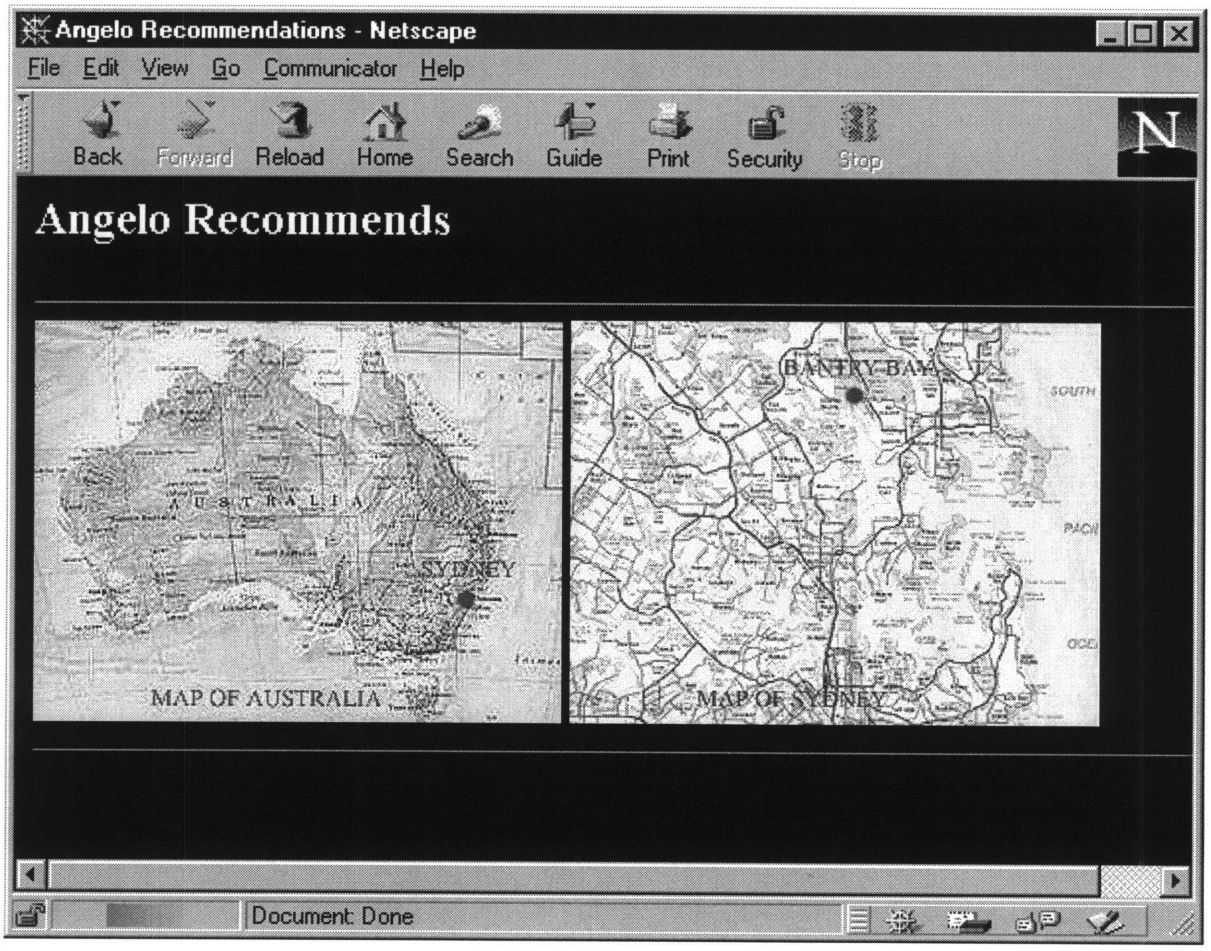

Figure 6.5. ANGELO adapts current page showing only maps.

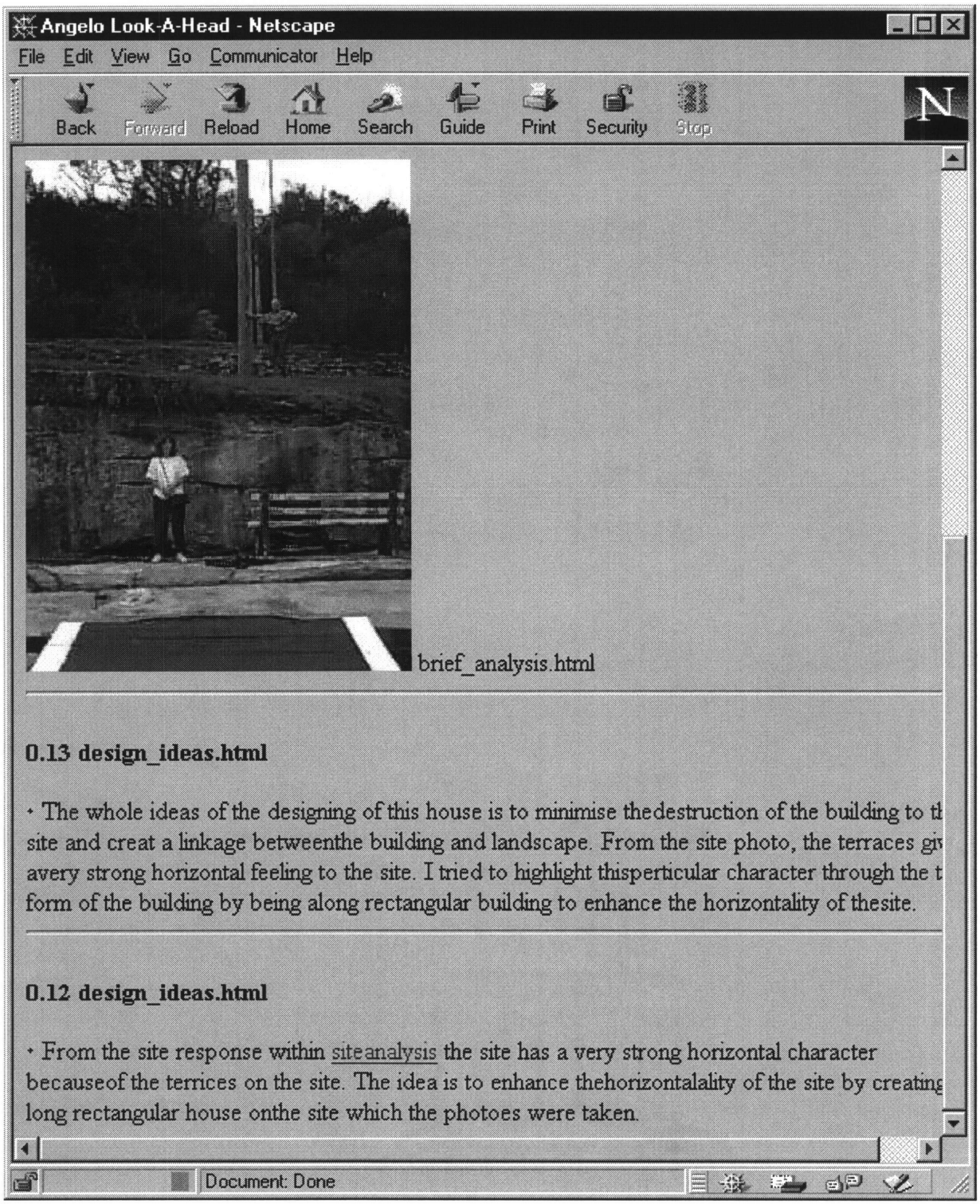

Figure 6.6. ANGELO look-ahead window.

7 Conclusions and Future Work

The thesis presented ANGELO as a contribution to the field of collaborative environments using software agents for assisting users with information overload and customizable information search. It builds a user profile in such environments in order to adapt and present the content according to the user's interest. Its functionality includes the usage of a combination of learning by example algorithm and information filtering/retrieval. In addition, the agent will assist in the documentation and the adaptation of presentation. Application of such system is to act as a front-end assistant for collaborative meeting documentation, assisting users to collaborate and negotiate better by presenting information from collaboration and meetings according to their preferences and view points.

The effectiveness of ANGELO still needs to be tested further with a larger database and vigorous user testing. Still, the implementation presented offered an initial prototype of how such a tool would help in collaboration process among professionals by giving individuals a better personalized and adapted interfaces to meetings.

Several open issues remain unanswered by our approach and several technical hurdles that will be subjects of future work. As expressed in Chapter 2, there are two requirements that the current implementation of ANGELO did not cover. The issue of modes of information presentation and the collaboration of knowledge learnt.

In taking the modes of information presentation in consideration, ANGELO would need to manage multiple user contexts and adapt the presentation accordingly. Depending on the time of the interaction, the user may be in a research mode, or a discussion mode, which would imply very different user models. Modeling these modes and allowing simple switching and combination of these user contexts is an important research issue for future work. The work can also be extended to allow the agent creates group memory and collaborate with other meeting agents in a computer-enabled collaborative environment.

Other extension of this work includes the automation of feature extraction from documents. This includes the concept extraction from images and other media. In addition, a better semantic understanding of the text is needed. Such latter extension to the Savant engine is to combine Word-Net (Miller, 95) as a back-end.

REFERENCES

W. Andrews, "Personalized Agents Enable Preferred Surfing", Web Week, Vol. 2, No. 12, August 19, 1996, Mecklermedia Corp.

N. Belkin, W. Croft, "Information Filtering and Information Retrieval: Two sides of the Same Coin?", Communication of the ACM, Vol. 35, No. 12, pp. 29-38, 1992.

D. Benyon, D. Murray. "Developing Adaptive Systems to Fit Individual Aptitudes," Proceedings of the 1993 International Workshop on Intelligent User Interface, pp. 115-121, 1993.

R. Bergman, "Learning structures in a manifest casual environment", Ph.D., Lab of Computer Science, MIT. 1994.

S. Bodker and K. Gronbae, "Cooperative Prototyping: Users and Designers in Mutual Activity", International Journal of Man Machine Studies, Vol. 34, No. 3, pp. 453-478, 1991.

A. Cypher. "EAGER: programming repetitive tasks by example," in Prod. ACM SIGCHI'91. pp. 33-40. New Orleans. May 1991.

FireFly Network Inc., http://www.firefly.com, October 1996.

D. Fisher, J.H. Generi, P. Langley, "Model of Incremental Concept Formation", Machine Learning: Paradigms and Methods, MIT Press, September 1989, pp. 11-62.

G. Furnas, et. al., "The Vocabulary Problem in Human-System Communication", Comm. ACM, Nov. 1987, pp. 964-971.

D. Grecu & D. Brown, "Dimensions of Learning in Agent-based Design", Machine Learning in Design Workshop, Fourth International Conference on Artificial Intelligence in Design, June 1996.

S. Greenberg, "Computer Supported Cooperative Work and Grouper: An introduction to the", International Journal of Man Machine Studies, Vol. 34, No. 2, pp. 133-143, 1991.

E. Horvitz, http://research.microsoft.com/~horvitz (February 97).

J. Kay, "Pragmatic User Modeling for Adaptive Interfaces", Adaptive User Interfaces: Principles and Practices, München, German, 1993.

R. Kozierok, "A Learning Approach to Knowledge Acquisition for Intelligent Interface Agents", SM Thesis, Department of Electrical Engineering and Computer Science, MIT, May 1993.

Y. Lashkari, M. Metral, and P. Maes, "Collaborative Interface Agents", Proceedings of AAAI '94 Conference, Seattle, Washington, August 1994.

M. Lea and R. Spears, "Computer-Mediated Communication, Dehumidification and Group Decision Making", International Journal of Man Machine Studies, Vol. 34, No. 2, pp. 283-302, 1991.

H Lieberman, "Mondrian: A Teachable Graphical Editor", in Watch What I Do: Programming by Demonstration, Allen Cypher, ed., MIT Press, 1993.

H Lieberman, "Letizia: An Agent That Assists Web Browsing", Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, August 1995.

H Lieberman, "Autonomous Interface Agents", Proceedings of the ACM Conference on Computers and Human Interface, CHI-97, Atlanta, Georgia, March 1997.

P. Maes, "Tutorial on Software Agents", Conference on Computer-Human Interaction, CHI 97. Also at http://pattie.www.media.mit.edu/people/pattie/CHI97/".

P. Maes, "Agents that reduce work and information overload", Communication of the ACM, July 1994.

D. Maulsby "Metamouse: "An Inscrutable Agent for Programming by Demonstration", in Watch What I Do: Programming by Demonstration, Allen Cypher, ed., MIT Press, 1993.

D. Maulsby, I. Witten, "Learning about Data from Examples and Hints". Technical Report. (1995), Department of Computer Science, University of Caldera, URL (June 96 @ fats// smi.stanford.edu/pub/Maulsby/).

T. Mitchell, "Personal WebWatcher: Implementation and Design", School of Computer Science, Carnegie Mellon University, Technical Report IJS-DP-7472, October, 1996.

A. Moukas. "Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem". To appear (1997): International Journal of Applied Artificial Intelligence.

J.F. Nunamaker et al., "Electronic Meeting Systems To Support Group Work," Comm. ACM, July 1991, pp. 40-61.

F. Peña-Mora, et. al, "Da Vinci Agent Society Initiative: Computer-Supported Negotiation Across Space and Time for the Life-Cycle Development of Sustainable Large-Integrated Engineering Systems in a Collaborative-Competitive, Domain-Dependent, and Strategy-Influenced Environment". MIT I.E.S.L., Technical Report. No. IESL 96-03.

F. Peña-Mora, R.D. Sriram, R. Logcher, "Conlict Mitigation System for Collaborative Engineering", Artificial Intelligence for Engineering Design, Analysis and Manufacturing,

No. 9, 1995, pp 101-124.

J.R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers Inc., 1993.

B. Rhodes, T. Starner, "Remembrance Agent: A continuously running automated information retrieval system", The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and MultiAgent Technology (PAAM '96), pp. 487-495.

Russell, "Artificial Intelligence: A Modern Approach", Addison Wessley, 1995.

G. Salton, M. McGill, " Introduction to Modern Information Retrieval", McGraw-Hill, 1983.

B. Sheth, P. Maes, "Evolving Agents for Personalized Information Filtering", IEEE Conference on Artificial Intelligence for Applications, 1993.

B. Shneiderman, "Taxonomy and Rules for the Selection of Interaction Styles", ACM CHI 95.

C. Stanfill, D. Waltz, "Toward Memory-Based Reasoning", Communication ACM, Vol. 29, No. 12, Dec. 1986, pp. 1213-1228.

J.W. Sullivan, S.W. Tyler, "Intelligent User Interfaces", ACM Press, N.Y., 1991.

S. Wu, http://www.arch.su.edu.au/kcdc/design_studio/students/design/wu_c/, International Virtual Design Studio 1995, Key Center of Design Computing, University of Sydney.