INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

A Software Tool for Providing Suggestions for Call Management

Imad-Roland Younes

A Major Report

In

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

June 1997

© Imad-Roland Younes, 1997

National Library of Canada

Acquisitions and Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada

Acquisitions et services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-40237-1

Canada

ABSTRACT

A Software Tool for Providing Suggestions for Call Management

Imad Roland Younes

Telecommunication is fast changing. New products and services are constantly introduced. Due to competition in this field, the time gap between the conception of a product/service to its deployment should be cut as small as possible. In addition users do not like to make to much effort to learn about new product/services. A software agent that can monitor the actions of a user and make proactive suggestions would be desirable. It is in this context, the present report is developed.

One of the problems in the use of agents in Telecommunication is: "How to offer "appropriate" proactive suggestions?" For a suggestion to be appropriate, it should be consistent with the previous actions taken by the user, the suggestion should be event-based, and it should be timely-based. A software tool is developed as part of this major report so that it can be used in two different ways in solving this problem. The first way is for researchers to simulate an agent's competency for offering appropriate suggestions to end-users. The second way is for researchers to demonstrate to vendors how an agent can develop proactive suggestions to be offered to users.

The goal of the software tool developed in this report is for researchers to study different strategies in offering appropriate suggestions to end-users, and demonstrate their outcomes to potential vendors who might be interested in such a product. The software tool is called FATMA (Futuristic Automated Telecommunication Management Agent) and is implemented using CLIPS, C, and TCL/TK.

Acknowledgments

Firstly, I would like to thank both my supervisors Dr. Clifford Grossner for helping me in my project and helping me writing my report, and Professor T. Radhakrishnan for all the ideas he gave me for my project and all the help for writing my report.

Special Thanks to Dr. Pardo Mustillo for the help and support throughout my project. I would also like to acknowledge Sasha Lenir, Steve Shepherd, Tu Nguyen, Jean-Pierre Rabbath, Denis Mcgonigal, and Luiza Solomon for the discussions and ideas that we had throughout the project.

Special thanks go to my uncle, Albert Younes, who provided me with his house to stay for all of the period of my Master Program. I appreciate and would like to thank my family for their financial and moral support, and strength. Finally, I would like to thank my Aunt, Ferial Younes, for her moral support throughout my Master program.

To the memory of my

Grandparents

Maroun Habib Younes 1909-1989

Hana Laurence Younes 1909-1996

Table of contents

List of Figures

List of Tables

1 Introduction

Whether at home or at work, people are relying more and more on computers and telephones for their daily tasks. Many of these tasks involve interactions between humans and machines. Some of these tasks require repetitive activities such as call forwarding, checking voice mail, searching for specific interests on the web, scheduling meetings between participants, incorporating and filtering e-mails, etc. Many researchers have tried to facilitate the performance of these tasks, by having machines assist humans.

One solution for relieving the user from performing repetitive tasks, is to develop a software program that automates the repetitive tasks on his or her behalf. These software programs should be somehow autonomous, in the sense that they control their own actions as opposed to direct manipulation by the user[13]. These software programs are called "intelligent agents."

Many researchers defined the word intelligent agent [4, 6, 7, 8, 9]. For our purposes, the definition of the word intelligent agent would be that intelligent agents are autonomous software programs that can achieve some of the user's tasks, without his or her intervention. In the last decade, many researchers have worked and focused on the issues of intelligent agents. The purpose of these studies were either to create agents that have a different task to achieve, or to design agents themselves. Such research lead to different types of agents. These agents could be categorized into the following categories:

1. Interface Agents: An interface agent is a "semi-intelligent", "semi-autonomous" agent that assists the user in dealing with his or her repetitive tasks. An interface agent is like a personal assistant that learns about the tasks, behavior, and preferences of its user, and uses what it learns to act on the user's behalf[4]. They are called semi-intelligent because they learn only the repetitive tasks of their user. If each situation is different from all prior ones, the agent would not be able to assist the user in automating his or her tasks. Furthermore, the agent is semi-autonomous because it automates only the tasks assigned to it from the user. Examples of interface agents include work by Kozeriok on an interface agent to help the user in scheduling his or her meetings[4]. Caglayan et. al. developed Open Sesame! for Apple Macintosh. Open Sesame! is an interface agent that learns high-level user events and automates these events[9]. Ali-Ahmad and Velasquez worked on "E!Agent", they extended a user interface agent's learning and behavior through emotion perception for e-mail filtering[10].

2. Networked Agents: Networked agents are intelligent agents that are connected to a computer network. Networked agents provide an intelligent interface to the user and make extensive use of the various services available in the network[8]. A networked agent can access local as well as remote resources for achieving its tasks. A good example of networked agents are "search engines." Lieberman introduced Letizia, an agent that assists in Web Browsing[11]. Letizia will search for documents and web sites that could interest the user and recommend these sites to the user.

3. Collaborative Agents: Collaborative agents are multi-agent systems that coordinate with each other to achieve a specific task[14]. Collaborative agents coordinate with each other by sharing their knowledge, goals, and skills to jointly take actions or solve

problems[8]. One type of collaborative agent, is where agents communicate with each other in the form of request and reply messages. An example of this type of collaboration, is Lashkari et. al.'s e-mail agent where agents of different users communicate with each other, so that every agent will learn the behavior of its user faster than by only observing the actions of its user[12].

4. Mobile Agents: Mobile agents are intelligent agents that travel in large computer networks offering a number of sophisticated services. Examples are ranging from advanced Internet filter and search agents, smart messaging, to intelligent communication and management. They may communicate with the user, the services available in the network, and other agents[8]. Mobile agents travel through the network in a pseudo-connectionless fashion. When an agent navigates in the network to reach its destination, the agent does not keep track of the path it took. However, they can find their way back to their owner, if the assigned task requires so[13].

In this paper, we will focus our work on interface agents. Other types of agents will not be discussed in this report.

1.1 What is a Proactive Agent?

Reactivity and Proactivity are two important characteristics in an interface agent. An interface agent is reactive if it perceives the changes in its environment and responds to those changes. An interface agent is called proactive if it takes initiatives to achieve goals[13]. A proactive agent, not only reacts or responds to the user's needs, but also takes initiatives and acts without the interference of the user when the agent needs to do so.

Many techniques can be used in order for a proactive agent to assist the user in his or her tasks and applications. One is the knowledge-based approach suggested by Kaye and Karam[2], where a knowledge engineer programs the agent in advance with a comprehensive set of rules. In this approach, the agent is preprogrammed with a set of rules that are invoked to automate a particular task. Another is the user-programming approach suggested by Malone and Lai[3], where the user programs the agent to perform the tasks he or she wishes to have automated. In this approach, the user has to learn how to program the agent. There is a third approach which is more flexible than the knowledge-based approach, and does not require the user programming expertise. This approach is called the machine learning approach.

> "The Agent learns the user's behavioral patterns by observing the user's interaction with the application. When it is "confident" that it knows what the user would do in a situation, it can automate that action for the user, by interacting with the application in the ways it has observed the user doing so. It can receive feedback from the user when its prediction is incorrect."[4]

In this learning scenario an algorithm is necessary whereby the agent is able to determine if it is confident enough to act on behalf of the user. This learning approach is called Memory-Based reasoning[5].

In order to have a proactive agent, the agent must satisfy the following features:

Learning: In the learning process, the agent must observe the actions of its user while the user is performing a certain task. The agent must then remember the actions and behavior of its user when performing this task. One way for remembering the behavior of the user is

for the agent to store the required data in a database. The agent must also have knowledge about the features of the task the user is performing. This knowledge could be acquired, either from a knowledge engineer who instructed the agent about the features required, and/or learned by the agent when observing and remembering the behavior of its user. Then the agent must be able to apply what it observed and remembered in performing similar tasks.

Trustworthy: Another aspect for a proactive agent is to be able to show how trustworthy it can be. The user must feel confident about the actions of his or her agent and feel comfortable assigning tasks to it. In order for the agent to show how trustworthy it can be, it could interact with the user to either ask questions on how to act for a particular task[10], or to offer recommendations and suggestions[11]. The user will then see that the agent will not act unless it is certain of its own actions, and so the user will feel confident about his or her agent actions. The interaction between the agent and the user leads to fewer errors in the agent's actions because the user is helping the agent learn his or her behavior, and this will lead the user to gradually trust his or her agent and feel comfortable assigning tasks to it.

Helpful: The agent becomes helpful to the user only when it starts acting on his or her behalf. The agent should be consistent with its action in terms of what the user is trying to achieve. In other words, the agent should only perform a specific action when this action needs to be done, thus relieving the user from performing it, and saving the user's time. The user will then feel that having his or her agent was worthwhile because it is helping him or her in performing some of his or her tasks.

1.2 How to Offer Proactive Suggestions?

One of the problems in the use of agents that are proactive and learn in telecommunications is: "How to offer "appropriate" proactive suggestions?" For the suggestions to be appropriate, they should satisfy the following three criteria:

- Suggestions should be consistent with the actions the user has taken in the past. The agent should offer a suggestion to the user to automate a particular task if the agent observed the user *consistently* performing that activity in the past. Sometimes offering a suggestion for a particular tasks shows that the agent is less intelligent, if the user never or *rarely* uses this task.
- Suggestions offered should be event-based[1]. The suggestions will be appropriate only if they are relevant to the context of what the user is trying to do. The agent should not interrupt the user to offer a suggestion about a particular task when the user is performing a different type of activity. Otherwise, the agent will be interrupting the user at a time where he or she is busy working with other activities and might confuse him or her.
- Suggestions offered should be timely-based[2]. The suggestions will be appropriate only if they are offered at an appropriate moment in time. If the user is working on a particular activity, the agent should not offer a suggestion to perform this activity if its suggestion is at an inappropriate time.

[1] Event-based means that the suggestion offered depends on the event arising out of the user's action.
[2] Timely-based means that the suggestion offered depends on a certain moment in time. actually, for a suggestion to be offered, event-based and time-based criteria should be satisfied.

As a component of my research, a software tool is developed so that user interface designers can study different solutions for determining the criteria used to ensure that a proactive agent offers suggestions that are appropriate. To enable the development and testing of criteria for offering appropriate suggestions, I developed FATMA (Futuristic Automated Telecommunication Management Agent). For FATMA to efficiently serve its purpose, and efficiently help researchers in studying solutions for offering appropriate suggestions, I developed a set of observables that FATMA uses to learn the behavior of the user. I also developed sets of rules to activate the proactive functionality for FATMA to offer suggestions, and to demonstrate how FATMA works.

Wildfire is a telephone voice assistant that manages incoming and outgoing telephone calls, contact lists, and voice messages[1,15]. Wildfire can take messages, check who is calling before passing the call to the user, forward calls to other remote locations, etc. FATMA resembles Wildfire in that FATMA is a testbed built to help the user with his or her **call management** tasks, with the capabilities for it to offer suggestions to automate the user's repetitive tasks. In doing so, the user will benefit by being relieved from performing some repetitive tasks when managing his or her calls.

1.3 Organization of this Report

Chapter 2 discusses various studies on the design of interface agents, how to build interface agents that help their users in an easy and friendly way. Chapter 2 also discusses other related interface agents, the problems encountered with these agents, some attempts to solve these problems by either changing the learning approach or by extending the existing learning approach, and some lessons learned from these problems. Finally, chapter

2 discusses some command driven SUI (Speech User Interface) agents used for telecommunication services, and some problems encountered with these agents.

In Chapter 3, I introduce FATMA. I describe the way in which researchers can use FATMA, I state the feature sets and proactive functionality that FATMA provides, and the way FATMA observes the behavior of the user. I also describe FATMA's system architecture; and I describe the rule sets I developed to test and demonstrate how FATMA works. Finally, I describe the scenario I built, and the demonstration I did to evaluate my study.

Chapter 4 provides a short summary and conclusion of the paper. It also discusses some future work to enhance and extend the functionality of FATMA, and some ideas for more studies on how can FATMA offer appropriate suggestions.

2 A Survey of Related Work

In the last ten years, there has been a growing interest in the design and creation of software agents. Several researchers have focused on the issues of intelligent agents[7, 8, 16]. Some of the studies were to improve the design of the agents, and others were to build agents for achieving specific tasks. The desirable characteristics of an agent varies widely for it to be useful and acceptable to the users. Kautz et. al.[16] believe that agents must relieve the users from low-level tasks. Many users don't like to spend time working on low-level routine tasks to achieve their objectives. Agents could assist users in performing mundane and routine tasks. Users must feel that having an agent to assist them is worthwhile. Another aspect Kautz et. al. address is that agents should be as much friendly and easy to use as possible, because people don't like to change their work habits.

In his paper "Attaching Interface Agent Software to Applications"[7], Lieberman discussed the idea of how to include software agents into existing applications. Attaching agents to existing applications aimed to solve two problems. The first problem is to deal with the reluctance of users to change. Attaching an agent to an existing application minimizes the amount of learning in using a new application; the agent will adapt to the user's behavior and relieve him or her from performing repetitive tasks, instead of having the user adapt to the new features. The second problem addressed by attaching an agent to an existing application, is that the researcher, designer, or agent would not have to create

a new application just to test his or her agent. Besides, with commercially available applications, the researcher will be able to test his or her agent with more realistic features.

Having the above characteristics in mind, many researchers have built software agents to assist in specific tasks such as, e-mail filtering, performing administrative and clerical tasks, searching for "interesting subjects" in news-groups, searching for interesting documents and web sites on the Internet, meeting scheduling, supporting interactive features in telecommunication, etc. With the design of these agents, new problems arose. In the next section, I discuss some of the tasks which interface agents perform; what problems they encounter; and some solutions to those problems. In section 2.2, I discuss several telecommunication interface agents; and what problems they encounter. Section 2.3 concludes this survey.

2.1 Problems with interface Agents

Kay and Karam developed a cooperating knowledge-based agent for the office[2]. Their system called COKES is a knowledge-based office agent aimed to support several concurrent activities. It is also integrated with a computer-based message system (CBMS). In their work a "consultation" represents a high-level office task. A consultation begins with some target goal that is to be achieved, which may originate from human operator or from another office agent. The office agents use a conventional passive CBMS to exchange two types of active messages when communicating with each other: *consultation requests*, and *knowledge transfer*. A consultation request is a request for the message receiver to carry out a specific consultation. Knowledge transfer via message passing is used to send knowledge from one office agent to another.

COKES' knowledge is based on logic obtained from knowledge engineers. The knowledge engineers preprogram the office agents so that the agents can cooperate and solve problems independent of their users, whenever appropriate. Although the system, in some cases, is capable of reasoning, acting, and clarifying its own actions, the problem of flexibility remains. The actions that the agents perform are dependent on the preprogrammed rules. Since their rules are preprogrammed, the agents do not observe the behavior of their users, and they do not adapt to their users' behavior and performance. Another problem for the knowledge-based approach is that the agent's ability is determined by the knowledge engineer's expertise in predicting all the situations that an agent might find itself in. This type of approach would be helpful when the applications that the agent is solving are well defined. However, this is not always the case; many problems are complex and not fully defined, and the knowledge-based approach will fail in solving such problems because the user's behavior cannot be fully known a priori. Furthermore, the user will not be able to trust the agent because he or she does not know how the agent would perform when faced with an unknown situation.

Lai and Malone[3] took a different approach on how an agent could relieve the user by performing some tasks on his or her behalf. They built "Object Lens: A "Spreadsheet" for Cooperative Work", a user-programming approach where already built rules are *visible* and *changeable* by the user. "That is, users must be able to easily see and change the information and the processing rules included in the system." Furthermore, the user could also create his or her own rules so that the agent will automatically perform the given tasks. In this approach, the user definitely trusts his or her agent since he or she built the rules and usually expects the agent to perform the appropriate actions. However, the

agent's ability to act, depends on the user's ability to program the rules. The user must first find time to learn how to program the rules, and then find time to program them correctly. Furthermore, the user must be an expert in knowing and understanding all the system tasks as well as the programming rules, otherwise many of the rules he or she builds will not be very helpful in solving complex problems.

To solve some of the above problems, Kozeriok[4] considered the machine learning approach to design her meeting scheduling agent. She used memory-based reasoning and rule-based induction as the approach for learning the user's behavior. In this approach, the agent is flexible because it gains its knowledge by observing the user's actions instead of having all its knowledge provided by a knowledge engineer. The agent will gradually learn the user's behavior in performing certain actions and when it reaches a given level of confidence, it will start acting on his or her behalf, relieving the user from performing those actions. The agent here is *reactive* because it perceives the changes in the user's behavior, and responds to those changes, making it adaptable to the user's behavior. Moreover, the user will gradually gain trust in his or her agent because the actions of the agent depend only on the user's prior actions. The explanations provided by the agent could validate that its actions depend on the user's past performance. In the machine learning approach, the user does not need to learn how to program the agent, instead the agent learns and observes how the user behaves for it to act like the user. In this approach, the agent is more helpful than both knowledge-based and user-programming approaches since the agent will not act in a contradictory way to the user, and the user does not take a lot of time to teach the agent.

One problem in Kozeriok's approach is that the agent needs a sufficient amount of time to learn and observe the user's actions before it can be of any use[12]. Another problem is to decide when should the agent offer its suggestions to the user. Although the suggestions are consistent with the past actions of the user, these suggestions may not be event-based suggestions. The agent does not know what the user is performing right now. If the user is working on something not related to the suggestion offered by the agent, the suggestion offered could confuse the user and may not get the right feedback from him or her; the suggestion could also be annoying considering that the user may be busy doing other work and not wanting to be disturbed.

Lashkari et. al.[12] tried to solve the problem concerning the elapsed time required for the agent to learn by building a collaborative interface agent. Agents communicate and coordinate with each other to learn more quickly the user's behavior and act on their behalf. Lashkari et. al. applied their study to an e-mail filtering system and used the machine learning approach to supply the learning behavior of their agent. They also extended this learning approach by building an interface where the agents could communicate with each other by asking questions on how to act for a certain task. The agent's first approach for learning, is by observing the user's actions. "When faced with an unfamiliar situation, an agent consults its peers who may have the necessary experience to help it." When consulting its peers, the agent gets predictions only from agents that have sufficient experience on how to act for this particular behavior. The agent then, accepts one of these predictions and suggests it to the user. If the suggestion is rejected by the user, the agent will disregard future suggestions from the agent that gave it this incorrect prediction. With this approach, the agent starts narrowing down the agents that might help

it in the future. Furthermore, the agent will learn faster and can start acting on the user's behalf more quickly than a non-collaborative agent. However, using other agents' predictions is not always a good solution. The agent might get a good prediction from one agent for a certain situation and a bad prediction from the same agent for another situation. An agent which had been disregarded in the past, due to a bad prediction, might give a good prediction for another task, if it is reconsulted. (But it will not be reconsulted). Although the agent learns faster in this approach, many of its predictions may be incorrect and could annoy the user and make him or her lose trust in the agent.

Ali-Ahmad and Velasquez[10] took another approach to address the elapsed time taken by an agent to learn its user's behavior. To test their concept, an agent "E!Agent" was implemented. They considered MAXIMS, (an existing e-mail agent that was implemented at the Media Lab at MIT) which uses Memory-Based Reasoning as its learning algorithm. They extended this learning algorithm to include the "emotional state of the user". In this context, the agent would rely not only on repetitive tasks performed by the user for learning purposes, but also would observe the user's emotions. E!Agent's emotion learning process was based on the user's spoken words when receiving or filtering an e-mail. E!Agent needs a speech recognition system for it to understand what the user is saying. The agent determines the user's "situation" through a keyword analysis and by interacting with him or her to ask about what caused the situation. Although in this approach, the agent is shown to learn faster, the behavior of a user depends very much on the emotions. The same emotional state of the user does not always lead him or her to take the same actions. For example, receiving an e-mail telling the user that he or she was granted a job in a certain company, could result in the same emotional state as when

receiving a very funny joke in an e-mail. In both cases the user might be *happy*, but in the first case, the user may want to reply to the sender, while in the latter case, the user may re-file the e-mail. Another problem is that speech recognition and natural language have to be flaw less for this approach to work.

Lieberman[11] used the machine learning approach to supply the learning behavior for Letizia: an agent that assists the user in Web Browsing. Letizia uses the past behavior of the user to anticipate a rough approximation of the user's interests. Letizia does not take control of the user interface by acting on behalf of the user, it only offers suggestions for some documents and Web pages, when it finds them to be of the user's interest. Besides, this agent does not interrupt the user to offer a suggestion, but waits until the user explicitly asks for suggestions. This approach ignores the potentials of proactiveness of the agent; and if the user never asks or forgets to ask for any suggestions, a valuable suggestion found by the agent would be useless.

Unlike Lieberman, Caglayan et. al.[9] defined and discussed several issues encountered with proactive agents. The problems they showed were derived from actual customers' feedback from the commercially available agent Open Sesame!. Open Sesame! observes high-level user events in the use of general Macintosh applications and automates them. The agent uses machine learning to automate repetitive tasks performed by the user. The tasks are categorized into time-based and event-based tasks. Before the agent automates any new task, it first suggests the user that it can automate the task; the user could accept this, reject this, edit the suggestion to fine tune the task, or postpone the decision until a later time. The problems Open Sesame! encountered were more related to users interaction and people's high expectations for their agents. Many users found that

the agent has too few types of actions that it can automate. The users expected the agent to automate more actions than it normally handled. Other problems were the interactions between users and their agents. Some users claimed that interruptions for suggestions should happen less often, while others claimed that they should happen more often, but both type of users agreed that the agent's interruptions for offering suggestions should be controlled and prioritized. The agent should not interrupt the user any time it wishes, to offer a suggestion. Another problem that users noted, was that the agent could re-suggest automating a certain task even if the suggestion was already rejected by the user.

In table 2.1, I summarize related problems and solutions discussed thus far in six agents. It lists the advantages for using interface agents to assist users in their daily tasks. It shows the problems that occur while having an interface agent, and offers some solutions to those problems. Finally, it gives an idea of problems that are still open for future research.

Interface Agents		Problems or Shortcomings	Solutions or Advantages
Knowledge-Based Agent Approach (KBA)	COKES	-Do not observe nor adapt to the user's behavior. -Actions limited by the knowledge engineer expertise. -User do not trust the agent.	-Relieve the user from predefined tasks. -Clarify its own actions.
User-Programming Agents Approach (UPA)	Object-Lens	-Actions of the agent are limited to the user's expertise -Time consuming for the user to learn and program the rules.	-Relieve the user from user-defined tasks. -User edits and changes already defines rules. -User Trusts the agent.
	Meeting Scheduling Agent	-Agent needs sufficient amount of time to learn. -Suggestions are not time-based nor event-based suggestions.	-Agent observes and learn the user's behavior. -Relieve the user by automating repetitive tasks. -User trusts the agent. -The agent is more helpful than KBA and

			UPA.
	Collaborat-ive Interface Agents	-Agent may give a bad prediction to the user. -Many incorrect predictions annoy the user.	-Agents communicate with each other to learn faster and act on their users' behalf.
Machine Learning Agents Approach (MLA)	E!Agent	-Same predictions for the same emotional state could be incorrect. -Speech Recognition and Natural Language are not reliable.	-The agent learns faster the behavior of the user depending on the user emotional state. -The agent takes action depending on emotion and repetition.
	Letizia	-Agent is not Proactive	-The agent does not interrupt the user, it waits for the user to explicitly asks for recommendations
	Open Sesame!	-Agent has too few types of actions. -Interruptions should be controlled and prioritized. -Agent suggests already rejected suggestion. -When the agent offers time-based suggestion without the context of other events, makes the suggestion useless.	-Users like the automation of maintenance tasks that are easily forgotten. -Users like the ability to defer reviewing suggestions. -Users like the idea of editing suggestions.

Table 2.1 Summary for Problems and Solution with Various Agents.

2.2 Interface Agents for Telecommunication Applications

Modern telephones having built-in intelligence are far more than a passive communication device. Telephones can assist their users in their daily tasks like a human assistant does[15]. The number of "features" that telecommunication companies offer keep increasing and to make use of them or even choose the right feature for oneself is difficult[17]. Some features are easy to use such as the call display where the callee can see on a small telephone screen who the caller is; some other features are hard for the user to learn and remember how to use them: for example, for the user to hear the messages left on his or her voice mail, he or she has to remember at least two strings of numbers, the

first is the Dialing Number (DN) to connect to the voice mail, and the second is his or her Personal Identification number (PIN). Other hard features to remember are the "*" features. There are many such * features: can enable the user to check and call back the last number that called him or her, to make private calls, to disable the call waiting, to make a three way call, to forward his or her calls to another DN etc. In order to make use of many of these features, one could integrate them into a single service. This service could be made easy and friendly to select a feature and use it. In this section, I discuss two telecommunication services: The first is "Wildfire" which assists the user in his or her telecommunication tasks[18]. The second is "SpeechActs" which assists the user in some of the computer tasks such as checking and reading his or her e-mail, meeting scheduling between participants etc. through the telephone[21].

Wildfire is an interface agent that integrates most of the call management features that are provided by telecommunication companies. It makes use of speech recognition. As an electronic voice agent, Wildfire provides the user with many telephone-related tasks[18]:

- It announces to users their incoming calls,
- forwards incoming calls to a remote location, if necessary.
- maintains a "contact list" where the user can voice-dial his or her outgoing calls,
- schedules and reminds the user of his or her follow-up calls and action items,
- handles all the user's messaging, giving him or her voice-mail features,
- creates three way calls from any remote telephone.

The most important characteristic in Wildfire, is that the user can obtain all the above features using voice commands. The user does not need to remember the keystrokes to

use for each feature. The speech recognition component of Wildfire, makes the system more friendly and easy to use.

SpeechActs is a research project at Sun Micros System that allows users to telephone their Sun Workstation and interact with a wide range of Solaris applications, using voice input[19]. This project will allow users to read and filter their e-mail messages, voice messages, check for meetings and appointments and enter new meetings in their computer calendar, check the weather and stock quotes. With the mail application, users can hear their messages, reply to messages, skip forward and backward from one message to the next, and send a new message to any person whose name is on the user's list. The calendar application allows users to browse their own calendars as well as other users calendars[20]. The most important feature in SpeechActs is that the user can switch from one task to another in a single session.

Both, Wildfire and SpeechActs, are interface agents that assist the user in telecommunication oriented tasks with a Speech User Interface (SUI). One advantage of interface telecommunication agents such as Wildfire or SpeechActs is that such agents can assist the user at anytime anywhere because of the POTS. The user can call his or her agent from any telephone at any time and check for messages, place calls, check his or her e-mail, etc.[19]. With the rise of Telecommunication interface agents, such as Wildfire and SpeechActs, new problems arise:

- The first problem is to cope with less than perfect speech recognition and natural language processing. Both agents are SUI, so the interaction between the user and the machine depends on speech recognition and natural language[20]. SpeechActs researchers are working to solve such problems, but since speech recognition and

natural language are not in the scope of my project, I refer the reader to [19, 20, 21] for more information.

- Another problem is that researchers, as well as users, should differentiate between a SUI and a GUI[20]. First, when building a SUI the researcher should make the interaction feel conversational. The user feels that he or she is in conversation with another human-like-agent and not talking to a machine. At the same time, the user must be able to interrupt the system whenever he or she feels appropriate. With a GUI, the user could always cancel a task performed by the agent, while with a SUI "the synthesizer[1] is difficult to interrupt due to cross-talk in the telephone lines which prevents the speech recognizer from listening while the synthesizer is speaking"[20].
- Finally, with a GUI, the user could postpone feedback to the computer until later time, specially when he or she is dealing with windows, whereas with a SUI, the user can't control his or her feedback or actions if the agent is waiting for a certain action from the user.

Proactivity, as discussed in section 2.1, showed to be a very important characteristic for learning interface-agents. Without the proactive characteristic, the machine learning approach could become useless if the user does not ask for any suggestions or recommendations. In this context, a learning interface-agent must be proactive to meet its usefulness.

Wildfire and SpeechActs, as currently designed, are command-driven agents. Their functional capabilities depend on speech commands given by the user. Adding proactive functionality to these agents could improve the effectiveness of their interface in the

conventional telephony[17]. For example, if the agent takes an initiative to act on behalf of the user, the user will be relieved of many interactions that would otherwise be needed.

2.3 Conclusion

In the next chapter, I discuss how I designed and developed FATMA (Futuristic Automated Telecommunication Management Agent) a tool for researchers to study different solutions on how to offer appropriate suggestions at an appropriate time, making the suggestions event-based and timely-based.

Being proactive, interface agents face new sets of problems such as: the elapsed time they take for learning the behavior of the user and taking initiatives to act on his or her behalf, or to offer him or her suggestions. The proactive suggestions offered should be relevant to the context of what the user is trying to do and should be timely and useful. Lashkari et. al.[12], and Ali-Ahmad and Velasquez[10] have studied one problem in depth, that is the time required for an agent to learn before it acts. They have extended the machine learning approach to collaboration between agents, and learning to recognize the emotional state of the user.

[1] A synthesizer is what the agent use to talk or converse with the user.

3 Testbed for Call Management

The field of interface agents is in its infancy. As discussed in chapter 2, there are still a lot of problems that are worth looking at. As much as researchers want to relieve the user from performing repetitive tasks, researchers understand that the user does not want to spend a lot of time training his or her agent. On the other hand, researchers also want the user to trust his or her agent. These two problems caused the rise of the machine learning approach where the agent learns the behavior of the user, interacts with the user to offer suggestions which is expected to build a trust relationship[4].

The machine learning approach also has some problems: with this approach, it can so happen that the agent interrupts the user to offer suggestions without considering the task the user is performing, or it can take a "long" time for the agent to learn the behavior of the user. My research contributes to the field of interface-agents in that I designed FATMA (Futuristic Automated Telecommunication Management Agent) a testbed containing Call Management features. FATMA is intended to be used as a tool for researchers to study different strategies for offering "appropriate" suggestions. For the suggestions to be appropriate, they have to be consistent with the actions that the user has taken in the past; the suggestions have to be event-based in terms of being relevant to the context of what the user is trying to do; and the suggestions have to be timely-based. Furthermore, FATMA has the limitations of a SUI. The user cannot withhold the response when FATMA is waiting for an answer. i.e., although FATMA was designed with a GUI,

I took into consideration that it will be taking voice commands from the user as well as offering suggestions to the user in a speech form.

In the rest of this chapter, the word "researcher" defines the person who uses FATMA to study different strategies on how to solve the problem of offering appropriate suggestions. The word "user" (or end-user) defines the person who will be using the product derived from FATMA, for a telecommunication service.

FATMA can be used in two different ways. The first usage for FATMA, called "the simulation application", is for researchers to study solutions for how an interface agent can offer appropriate suggestions. Researchers can simulate any actions that the user normally makes when using the features provided by FATMA. Depending on the user's actions, FATMA observes certain behavior and offers suggestions as specified by researchers. The second usage of FATMA called "the Demonstration Application", is for researchers to demonstrate to potential vendors in detail how an interface agent observes, determines what suggestions could or are ready to be made, and how suggestions are offered, based on any predefined scenario that researchers create.

In this chapter, I will discuss FATMA, a tool to study solutions for offering appropriate suggestions. In the next section, I will describe how to use the testbed, I will present the Call Management feature-set provided by FATMA, the suggestions that could be offered by FATMA, and the observations made by FATMA. In section 3.2, I will discuss the system architecture of FATMA. Section 3.3 discusses the rules I developed to demonstrate the usage of FATMA. Section 3.4 will conclude this chapter by presenting the demonstration that I gave to evaluate the concept of my work.

3.1 FATMA

The Call Management features provided by FATMA are very similar to the ones offered by Wildfire: a commercial product described in [1]. Researchers can simulate the user placing calls through a voice dial, placing a private call, screening incoming calls by either taking the calls themselves, or having FATMA handle the call first, or directly send the incoming calls to voice mail, forwarding all calls to a different remote location, checking if there are any messages, etc. More details about the features provided by FATMA are given in section 3.1.2. While offering suggestions, the FATMA system follows two criteria: *frequency* and *consistency*. By the term *frequency*, I mean the occurrence of a certain task a relative number of times within a certain period. By the word *consistency*, I mean a repetitive action being performed routinely in the same manner. The suggestions provided by FATMA are discussed in section 3.1.3. To enable FATMA to offer suggestions, it must record the features requested by the user which I call Observables and discuss in section 3.1.4.

Figure 3.1 is a data flow diagram that shows the components contained within FATMA. The Observables are a set of facts that are recorded when the user requests a call management feature. The rule sets are designed by researchers to enable FATMA to offer appropriate suggestions. Based on the observables, these sets of rules will first try to match a pattern in the user's behavior, and asserts facts based on those patterns which are called Pattern Facts. Based on the Pattern Facts, the rules will assert new facts that might be used as suggestions later on to the user, these facts are called Pending Suggestions. Finally, Based on the Pending Suggestions, the rules will assert facts called Suggestions which will be suggested to the user. The rule sets will be discussed in detail in section 3.3.

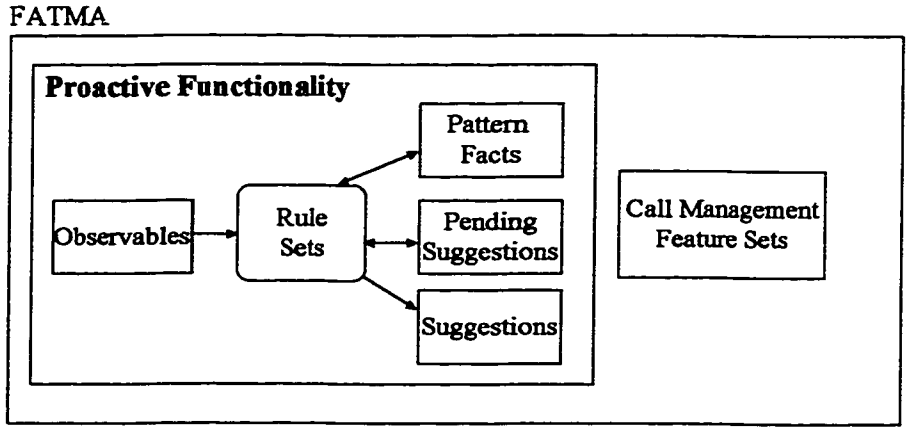

Figure 3.1 FATMA's Components.

3.1.1 FATMA's Usage

The Simulation Application

In the simulation application, several user actions can be simulated (see figure 3.2). Researchers can simulate the activity of a user placing a call, placing a private call, setting disposition for call screening, call forwarding, or checking his or her messages.

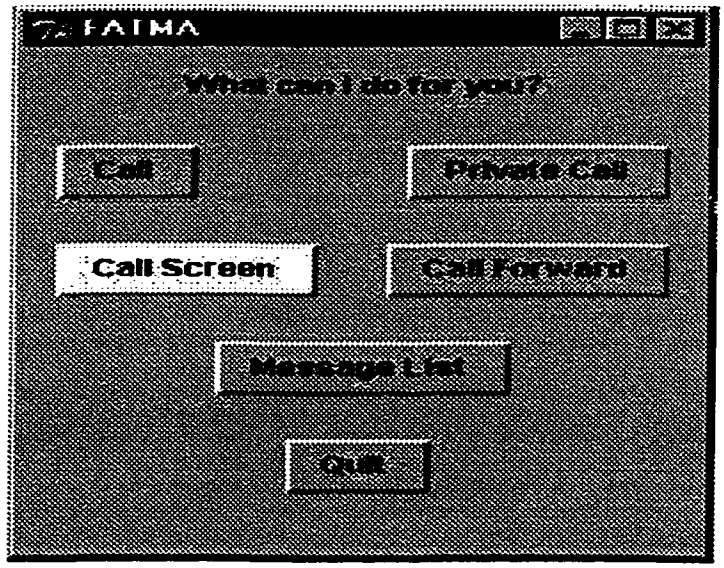

Figure 3.2 Main Window.

Depending on the feature selected, a second level window will appear. Figure 3.3 is an example of a call screening window where the user could set any screening

disposition to either an individual, or to a group of callers. When the user requests a feature and performs some actions in that context, FATMA will record the actions made by the user and depending on the feature, FATMA will store the data used to record the feature in an "observable". The data contained in the "observable" will be the name of the individual or group involved in the action taken by the user, the day, date and time the feature was requested and other data depending on the feature requested[1].

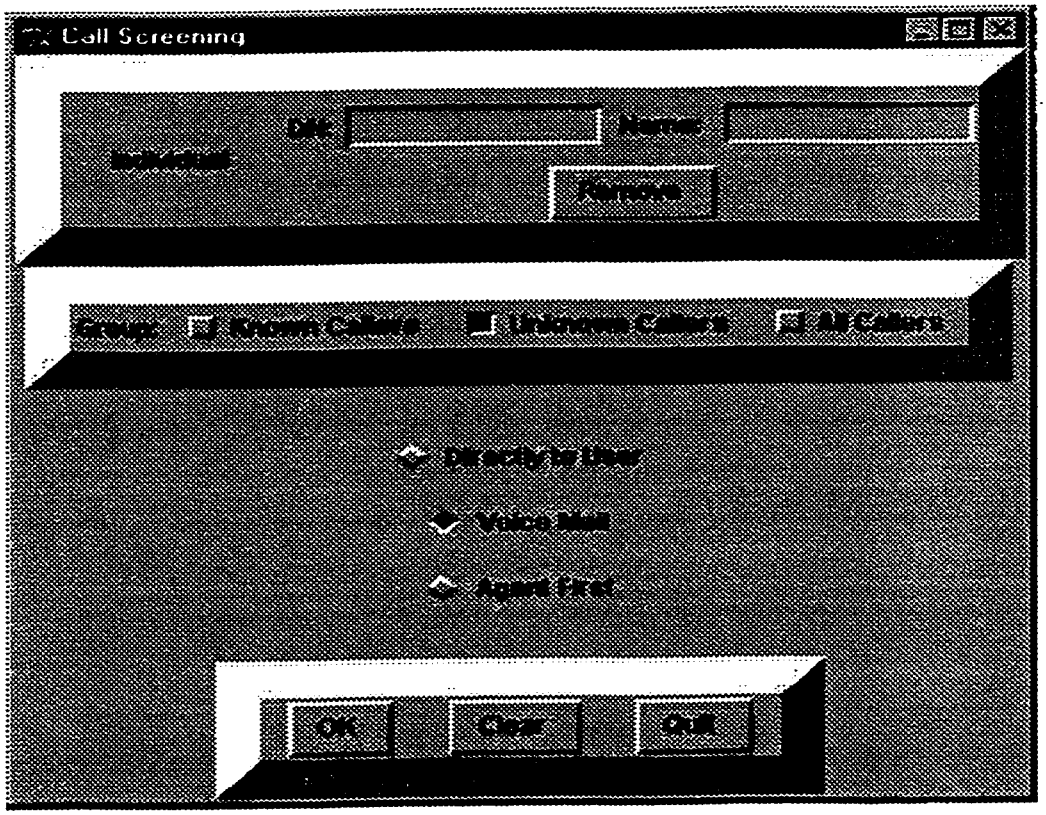

Figure 3.3 Call Screening Window.

When the user quits FATMA, the rules specified by researchers within FATMA are executed. The rules will take the observables as the basic input and asserts new facts such as the pattern facts or pending suggestions. Other rules in the rule set will determine which pending suggestions (if any) should be offered to the user the next time the user

[1] The data stored for every feature are described in section 3.1.4

requests a feature. The suggestions will be offered only if the user chooses the feature where a suggestion for that feature is ready. An example of a screening suggestion offered to the user is shown in figure 3.4 where FATMA is offering to set a group of callers to the voice mail for the weekdays from 9:00am to 5:00pm.

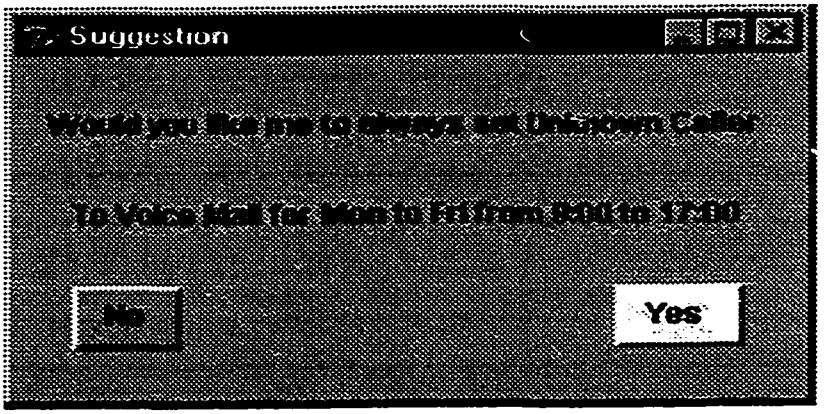

Figure 3.4 Suggestion Window.

The Demonstration Application

In this application, researchers can demonstrate to potential vendors in detail how FATMA records the actions of the user, how FATMA asserts pending suggestions, and how FATMA offers suggestions to the user specified by the researcher. For the demonstration application to work properly, it needs to have the simulation application running in the background, otherwise FATMA will not be able to offer any suggestions.

The demonstration application starts with a Control Window as shown in figure 3.5. In this window, the researcher will download different sets of observables to FATMA (up to three sets -the observables created from a predefined scenario- that will be shown in the Envelope Window (figure 3.6)). The Envelope Window shows the observables created by FATMA when a user makes a feature command from the simulation application, or by the researcher if the researcher is downloading a predefined scenario.

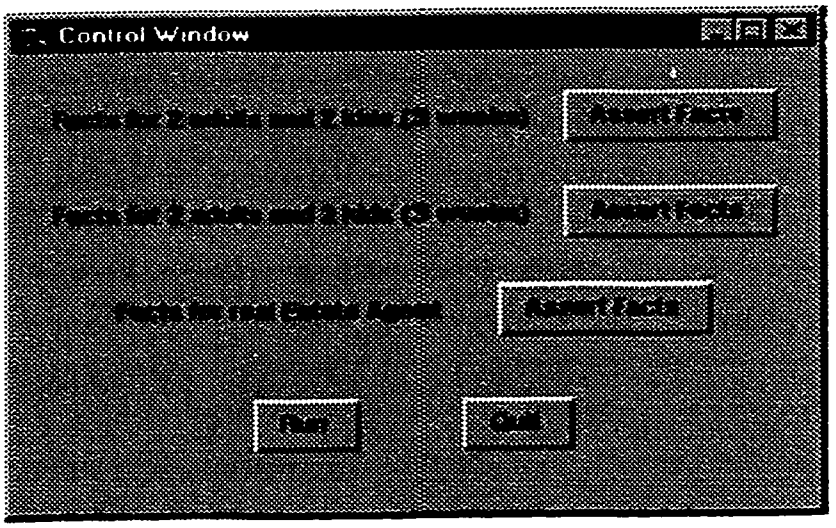

Figure 3.5 Control Window.

When the researcher runs FATMA from the control window with the predefined scenario, another window will appear, the Pending Suggestion Window (figure 3.7). The Pending Suggestion Window shows the suggestions that are ready to be offered when they are determined to be worth offering by FATMA through the rules. A pending suggestion is a fact that holds the following data: the name of the person or group involved with the feature, the days, starting time, and ending time of the feature performed by FATMA, and other data such as the disposition code for the call screening feature. After presenting the pending suggestions in the pending suggestion window, if the researcher chooses a specific feature from the main window of the simulation application (figure 3.2), FATMA will offer the suggestions related to this feature, if there are any.

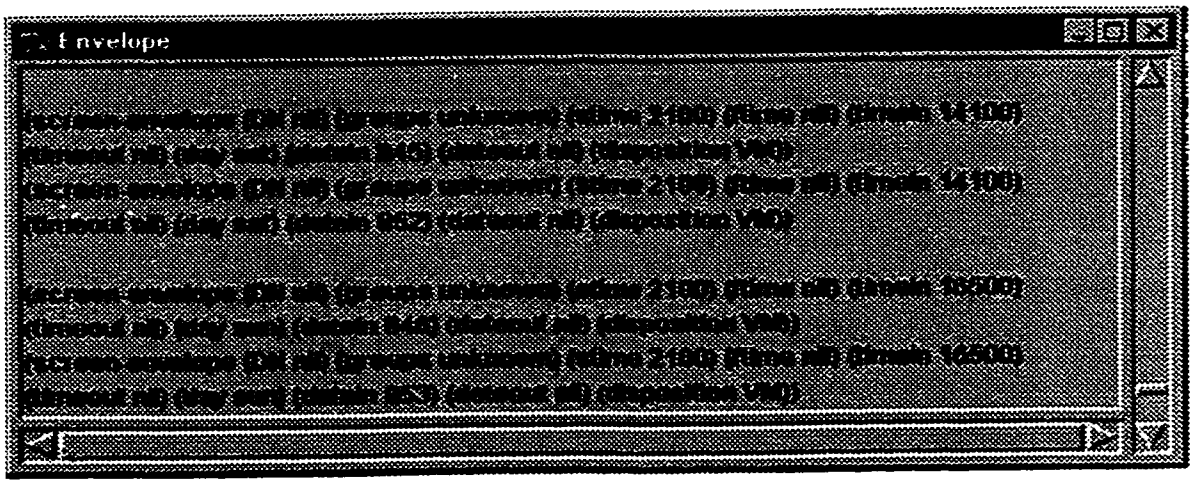

Figure 3.6 Envelope Window.

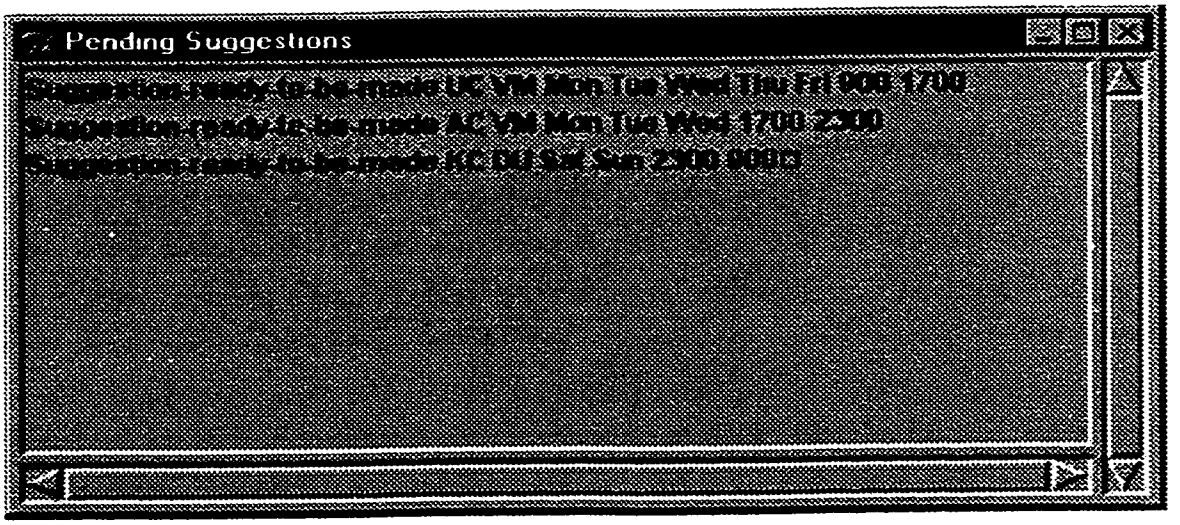

Figure 3.7 Pending Suggestion Window.

3.1.2 FATMA's Feature Set

The features provided by FATMA are the following:

- Contact List: FATMA has a directory where the user can store the dialing number (DN) of a person using the keypad, and store his or her name using voice. The user can add a name, delete a name, and review the list of names. The Contact List can hold up to 50 names. The user can call any DN in the contact list in 3 ways:
  - The user can dial the DN with the keypad.

- The user can voice-dial by saying "Call", then the name of the person as mentioned to the Contact List.
- The user can voice-dial by saying "Private Call", then the name of the person. With a private call, the callee will not be able to know who the caller is.

- Message List: FATMA has another directory where it stores the DN of the 10 last callers. If the DN of a caller is in the Contact List, the name attributed to the DN is also stored. A DN cannot be stored more than once. FATMA will store in the Message List the DN of the caller, his or her name if it is available in the Contact List, the number of times this DN called, and the time of the last call placed. The commands the user can give with this feature are:
  - The user requests this feature by saying "Message List".
  - The user can delete a DN from the Message List by saying "Delete".
  - The user can lock a DN in the Message List by saying "Lock".
  - The user can unlock a DN from the Message List by saying "Unlock".
  - The user can call back a DN from the Message List by saying "Call Back.".
- Call Screening: The incoming calls will be handled by FATMA in three ways depending on how the user instructed FATMA to select from numbers the three different disposition sets. The dispositions the user can give with this feature are:
  - Directly to User: The incoming calls will directly be handled by the user. The user sets this disposition by saying "Directly to User".
  - Agent First: The incoming calls will be handled by FATMA first, then either by the user or by the Voice Mail depending on the user's request. FATMA will handle the call and ask for the name of the caller if the DN is not in the Contact

List; then FATMA will inform the user who the caller is; finally the user will either take the call by saying "I'll Take it" or tell FATMA to transfer the caller to the Voice Mail by saying "Take a Message". The user sets this disposition by saying "Agent First".

- Voice Mail: FATMA will directly transfer the incoming calls to the Voice Mail. The user sets this disposition by saying "Voice Mail".

- Call Forwarding: The user can forward his or her incoming calls to any remote location such as work, mobile phone, etc.
  - The user requests this feature by saying "Call Forwarding", then the name of the person as mentioned in the Contact List, or the DN where the user wants to forward his or her calls to.
- Call Waiting: When a user is talking on the telephone and he or she gets a second call, the second caller will directly be handled by FATMA as if it is "Agent First" screening. FATMA will notify the user about the second caller. The user could either take the call or send it to Voice Mail.
- Groups: All the incoming calls are divided into 2 groups: the "Known Callers" are the callers that are on the Contact List, and the "Unknown Callers" are all other callers. An incoming private call is considered as an Unknown Caller even if the Caller's DN is in the Contact List. Call Screening could be made to all callers, or any group.
- Individual: An individual is a caller that could be a Known Caller or an Unknown Caller. Call Screening could be made to an Individual. For example The user could screen all calls to Voice Mail, while screens only an individual to Directly to User.

3.1.3 FATMA's Suggestions

FATMA's proactive functionality is designed to meet the requirement for researchers to study various solutions for the problem of offering appropriate suggestions to callers based on repetitive actions performed. Since the use of FATMA is to try to determine the criteria that can be used to ensure that its suggestions are appropriate, other type of suggestions are not integrated to FATMA.

The suggestions that could be offered by FATMA are divided to three categories: Outgoing Call, Call Screening and Forwarding, and Incoming Call Handling suggestions. There are two kinds of suggestions: the suggestions could be **always ask me**, where FATMA always needs to offer the same suggestion to the user for it to perform the task, or could be **ask me and do it**, where FATMA offers a suggestion once, if it is accepted, the action will always be performed by FATMA on behalf of the user. The suggestions that could be offered by FATMA are:

Outgoing Call

- FATMA suggests to add a DN to the Contact List. This suggestion is made on a **always ask me** basis. The user can add the DN suggested by FATMA to the Contact List permanently, temporarily, or could choose not to add the DN. Note that temporarily could be several weeks or months. (Alternatively, the user could say "temporarily", to which FATMA would then ask "for how long?", the user could reply "two weeks". FATMA then, could specify the beginning and ending dates.)
- If the user adds a DN to the Contact List temporarily, before removing that DN, FATMA reconfirms the removal of the DN with the user. The user could remove the

DN, or keep it permanently or temporarily. This suggestion is made on a **always ask me** basis.

- If the Contact List is nearly full, FATMA could suggest to remove a DN if that DN is *rarely* voice-dialed. This suggestion is made on an **always ask me** basis.
- FATMA suggests adding a DN from the Message List to the Contact List if the user calls back this DN, and if the DN is not already stored in the Contact List. . This suggestion is made on an **always ask me** basis.
- FATMA suggests removing the DN that the user normally removes from the Message List. This suggestion is made on an **ask me and do it** basis.
- If the user dials a DN that is already in the Contact List, FATMA could remind him or her that this DN is already in the Contact List, and therefore the agent could call on behalf of the user.
- FATMA could suggest always placing a Private Call for a certain DN. This suggestion is made on an **ask me and do it** basis.
- If a particular DN is locked for a certain period of time, FATMA could suggest adding this particular DN to the Contact List if it is not already stored there.

Call Screening and Forwarding

All screening suggestions from FATMA to the user are offered within the Call Screening feature.

- FATMA could suggest transferring a call from a particular DN as an Individual to "Voice Mail". This suggestion is made on an **ask me and do it** basis.

- FATMA could suggest to set the disposition to "Directly to User" to a call from a particular DN as an individual. This suggestion is made on an **ask me and do it** basis.
- FATMA could suggest to set the disposition to "Agent First" to a call from a particular DN as an individual. This suggestion is made on an **ask me and do it** basis.

The three suggestions mentioned above could also be proposed for Known, Unknown, or all callers. The actions for the suggestions mentioned above will be enforced for a *period of time*[2] set by either FATMA or the user. The actions for the suggestions mentioned above are set back to the original disposition set by the user, when their *period of time* is over.

- If the user forwards his or her calls to a particular DN, FATMA will suggest to automatically forward the calls to that particular DN for the user for a *period of time*. This suggestion is made on an **ask me and do it** basis.
- If the user calls FATMA from a particular remote location, FATMA will suggest to automatically forward the calls to the DN of that particular location for a *period of time*. This suggestion is made on an **ask me and do it** basis.

Incoming Call

The suggestions offered by FATMA to the user for the Incoming Calls are only made when the screening disposition is set to "Agent First" or when FATMA is handling a call waiting process.

[2] Period of Time starts when the user sets a screening disposition to certain incoming calls and ends when the user sets another disposition to the same incoming calls.

- FATMA will suggest transferring a call from a particular DN to "Voice Mail". This DN will be set as an Individual in the Individual table. This suggestion is made on an **ask me and do it** basis.
- FATMA will suggest to set the disposition for a call from a particular DN to "Directly to User". This DN will be set as an Individual in the Individual table. This suggestion is made on an **ask me and do it** basis.

The two suggestions mentioned above could also be proposed to Known, Unknown or all callers. The action of these suggestions will be set off as soon as the user changes the screening disposition, or when the call waiting process is over.

3.1.4 FATMA's Observables

The most crucial part for an interface agent to act on behalf of the user is its learning process. An ineffective learning process could lead to erroneous suggestions and actions on the agent's part, which in turn could lead to the user not trusting the agent's competence. The agent must be selective when learning, in terms of what data it should observe, as well as what internal structure it should consider when storing these data[22].

I divided the observables of FATMA into four observation sets: observables for Outgoing Calls, observables for Call Screening, observables for Call Forwarding, and observables for Incoming Calls. These sets are mutually disjoint. Within each set, FATMA should also be selective in choosing its facts to determine whether to offer a suggestion or not, for a certain observation. In other words, within an observation set, FATMA's rule sets for pattern matching, and pending suggestions for a certain behavior should only

select the facts that deal with that behavior and not all the facts of the observation set itself. The observables are as follows:

Outgoing Calls

- FATMA observes the DN dialed by the user, the date, day and the time at which the call is placed, and stores these data items in a database. The table holding this information is called Outgoing Envelope.
- FATMA observes the calls placed by a user to a DN.
    - For calls placed by the user when dialing a DN from the keypad.
    - For calls placed by the user from the Message List.
    - For private calls placed by the user.
- FATMA observes the DN that is deleted from the Message List by the user.

Call Screening

- FATMA observes the screening commands given by the user and stores these items in the Screening Envelope. The Screening Envelope will include the DN for Individual, group code for Known, Unknown, or all, starting date, ending date, day, starting time, ending time, and disposition code for the Screening Commands.
    - FATMA observes the call disposition set by the user for a particular DN as Individual.
    - FATMA observes the call disposition set by the user for Known, Unknown, or all incoming calls.

Note: The call disposition hold in the Screening Envelope could be: "Directly to User", "Agent First", or "Voice Mail".

Call Forwarding

- FATMA observes the date, the day, the starting time when the call is forwarded to a certain DN by the user, the ending time when the calls are forwarded back to the original DN, and the DN to which the user forwards his or her calls to, and stores these data items in the Forward Envelope.
    - FATMA observes the call forwarding set by the user to a particular DN.
- FATMA observes the locations from which the user contacts it (e.g. home, work, mobile phone, or any remote location.)

Incoming Call

- FATMA observes the DN, the time, the date, the day and the name of the caller associated with the DN if it is in the Contact List, and stores these data items in the Incoming Envelope.
- When the call disposition is set to "Agent First", or in Call Waiting process, FATMA observes the actions taken by the user in terms of "I'll Take it" and "Take a Message".

Note: The data will be stored in the Incoming Envelope only when the call disposition is "Agent First", or in a Call Waiting process.

3.2 System Architecture

FATMA's data is contained in two tables. The Envelope table is the table that represents the actions FATMA observed the user performing. The Pending Suggestion table is the table that represents the suggestions FATMA could possibly offer to the user. For a suggestion to be offered, the user should perform a task for there which already exists a particular pending suggestion, and the time and date the user is performing that task, matches the time and date for that particular pending suggestion.

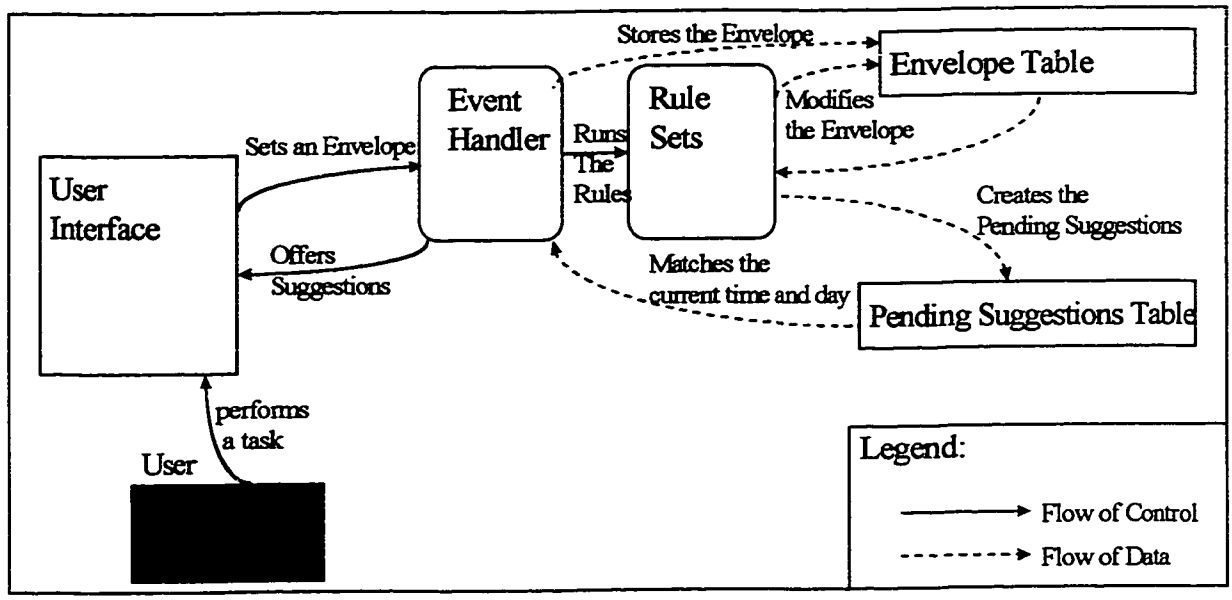

Figure 3.8 Simulation Application Diagram.

As shown in figure 3.8, at the implementation level, the main modules of FATMA are divided into three parts: the user interface, the event handler, and the rule sets. The user interface waits for an action to be performed by the user. The event handler is the module that receives from the user interface the event performed by the user, that adds to it the time, date, and day the feature that was requested, and stores it in the Envelope table. The event handler also takes from the Pending Suggestion table, the suggestions that are ready to be made, checks if they match with the current time and day; if the time

and day match together, the event handler gives the suggestions to the user interface to be offered to the user. The rule sets run on the completed envelopes in the envelope table to modify, change, or add pending suggestions to the suggestion table.

FATMA is intended to model a telecommunication service for Call Management, i.e. agents based on FATMA should reside in a telecommunication network, however FATMA itself is built to study solutions for offering appropriate suggestions to the user. The implementation of FATMA was carried out using an HP terminal as the user interface device, under UNIX operating system, using TCL/TK as the GUI, C as the event handler, and CLIPS as the rule based language. See figure 3.9: TCL/TK (Tool Command Language/Toolkit) as the GUI, researchers communicate with TCL/TK to perform specific tasks, and the GUI communicate with researchers to offer suggestions. Event handling was realized through programs written in C. They interact with the user interface either to take the command performed by the researcher, or to give the user interface the suggestions that should be offered to the researcher. The event handler takes the time and date from the Operating System time function, and also controls the firing of rules in the CLIPS rule base. CLIPS (C Language Integrated Production System) is the rule based language used for FATMA to perform the observation rules, pending suggestion rules, and the rules to determine which suggestions should be offered. CLIPS was embedded in C, i.e., all CLIPS command were given to it as C functions.

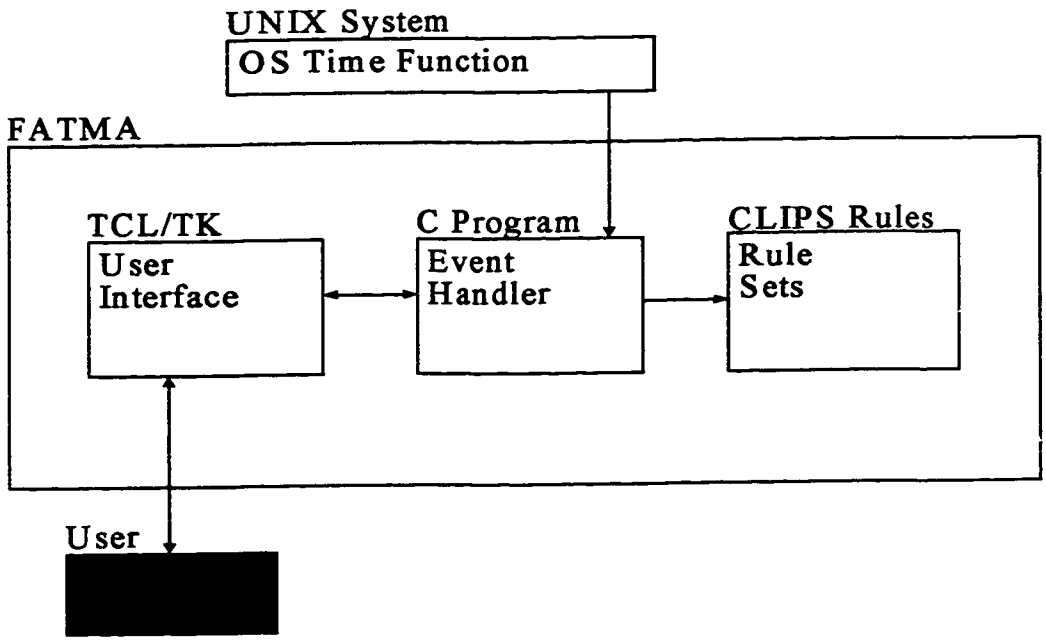

Figure 3.9 FATMA's Language Diagram.

3.3 Rule Sets for FATMA

In order for researchers to study solutions for offering appropriate suggestions, they should be able to create rule sets to activate FATMA's proactive functionality. When necessary, they should add new rules to the current ones, or change and modify the existing rules. In this context, the independence between the three modules: the user interface, the event handler, and the rule sets (see figure 3.9) is crucial because any change in one module should not cause a change in any other module. We aim at any modification in the CLIPS rules not leading to any modifications in the two other modules. Furthermore, CLIPS facilitates the development of software to model human knowledge, and being a stand alone rule base language, CLIPS facilitates the tasks for researchers to build, add, modify rules, and embed them in FATMA so that the researchers can evaluate their studies and demonstrate their solutions.

To be able to test and demonstrate how FATMA works, I developed rule sets for FATMA to offer "appropriate" suggestions for the "Call Screening" features it provides. The rule sets where divided into the following:

- The first set of rules called "Pattern Rules" are for FATMA to search for a *pattern* in the actions performed by the user and records them as Patter Facts.
- The second set of rules called "Pending Suggestions Rules" are for FATMA to determine from the Pattern Facts what are the pending suggestions.
- The third set of rules called "Offering Suggestions Rules" are for FATMA to determine, from similar pending suggestions, which pending suggestions should be offered and which should be disregarded.
- The fourth set of rules called "Re-offering Suggestions Rules" are for FATMA to determine if, which, and when a suggestion that was rejected by the user, should be re-offered.

Figure 3.10 is a flowchart that describes how the rule sets work to enable FATMA offer suggestions. This flowchart describes how the rule sets search and match a *pattern*. If a pattern in the user's behavior is found, the rule sets assert pending suggestions. The pending suggestion that best describes the user's behavior, is offered. If the suggestion is rejected by the user, it could be re-offered by FATMA at a later time if FATMA continues to observe the same regularities in the user's behavior.

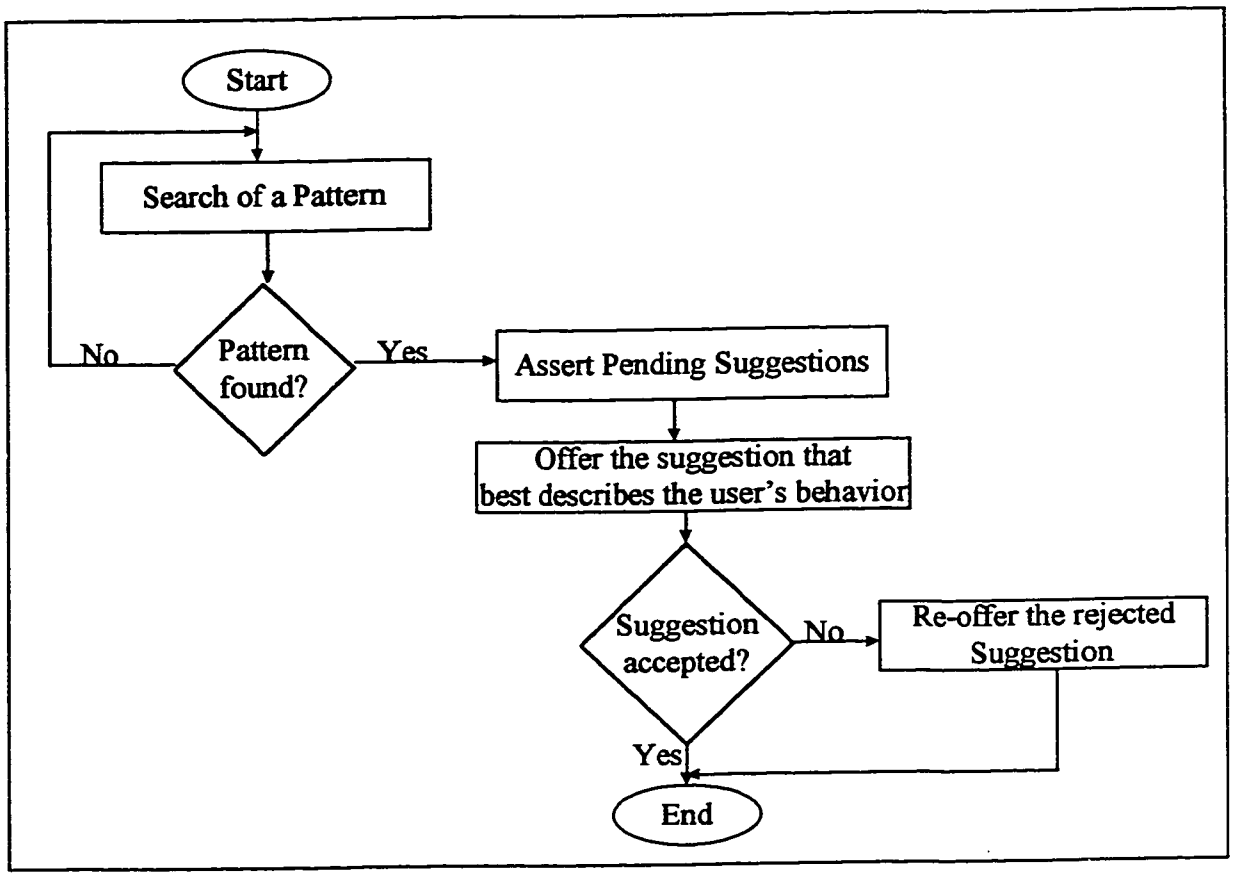

Figure 3.10 Flow Chart for the rule sets process.

In figure 3.11, I show the different rule sets and how they interact with each other. The pattern rules asserts facts that I call Frequency and Consistency facts, from which the pending suggestions rules determine what are the pending suggestions. The offering suggestions rules are the rules that determine from the pending suggestion table which pending suggestions should be offered and assert them as Offer Suggestions Facts.

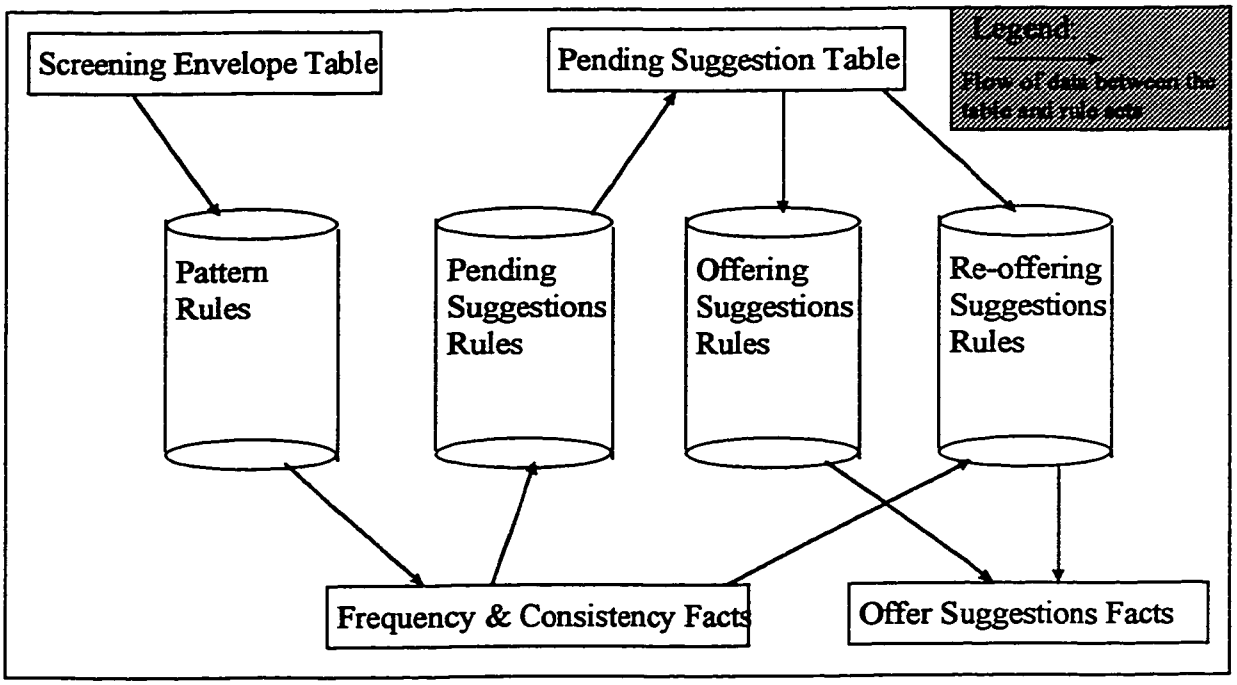

Figure 3.11 Rule Sets.

If there are any suggestions that are rejected, the re-offering suggestion rules will determine if these suggestions should be re-offered by checking the Frequency and Consistency facts and the Pending Suggestion table. If the same pattern is found the re-offering suggestion rules modify the Offer Suggestions Facts.

For a suggestion to be offered, the three modules are involved; the rule sets should determine the suggestions that could be offered, the event handler determines which suggestions match the current day and time, and the suggestion that is ready to be offered should match the feature used at the current time in the user interface. The rest of this section describes in detail the rule sets and the algorithms developed.

The pattern matching of FATMA totally depends on the observables given to FATMA. Thus, the Screening Envelopes are the basic facts for FATMA to offer suggestions. Since FATMA depends on regularity in the user's behavior, in the pattern

matching process and for simulation purposes, I assumed that users have weekly patterns. So in the pattern rules, FATMA performs two major sets of rules as shown in figure 3.12: the first set of rules called "Rules for Frequency" determines how frequently a feature is performed. The second set of rules called "Rules for Consistency" determines the features that are consistently performed on a proposed cyclic pattern. In the first set of rules, for FATMA to consider the same feature being performed frequently, the condition for that feature should satisfy the following criteria: the feature should be performed for a number of consecutive weeks on the same day of the week at approximately the same time for the same group of callers or individual. When the above condition is satisfied, FATMA will count the number of times this feature was performed (This condition will be referred to as C1 in figure 3.13). In the second set of rules, for FATMA to consider the same feature being performed consistently in the same week, the condition for that feature should satisfy that the same feature is performed to set the same disposition for the same group of callers or the same individual at approximately the same time for at least two days of the same week (This condition will be referred to as C2 in figure 3.13).

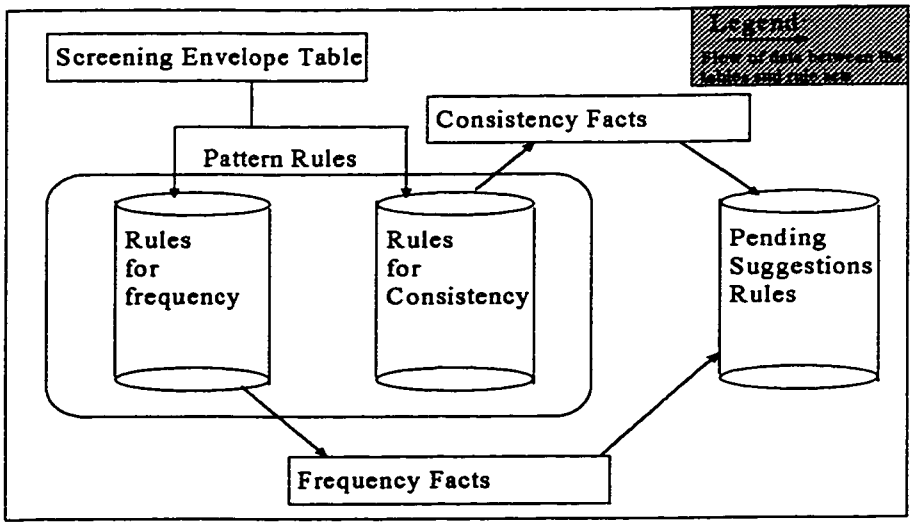

Figure 3.12 Pattern Rule Sets.

When the first condition (C1) is satisfied, it could lead FATMA to create a pending suggestion; but since I assumed a weekly pattern as the behavior of the user, to determine a pending suggestion only from the first condition, the occurrence of such a feature should be accomplished a "great amount" of time. On the other hand, when both conditions (C1, C2) are satisfied, I assume that the feature requested by the user is consistent with his or her behavior, thus a pending suggestion would be created with less occurrences than with only the first condition. The other conditions that should be satisfied for the agent to create pending suggestions are: The three conditions referred to as C3, C4, C5 respectively in figure 3.13 (1) are the same, these conditions should satisfy the two conditions (C1, C2) mentioned above and repeatedly performed for a threshold chosen by researchers. These conditions (C3, C4, C5) should satisfy the same disposition for the same group of callers or the same individual for at least three, four, and five days from the same week for at least three consecutive weeks on a row. The fourth condition referred to as C6 in figure 3.13 (2) should satisfy the two conditions (C1, C2) mentioned in the pattern rules by occurring two days in the same week for at least four consecutive weeks on a row. The last condition referred to as C7 in figure 3.13 (3) for creating a pending suggestion is the occurrence of the first condition of the pattern rules for at least six weeks on a row. In Figure 3.13 the markings (1) (2) (3) show what conditions should be satisfied for asserting pending suggestions. Note that a pending suggestion will not necessarily be offered. Later, we will examine which pending suggestions would be offered and which would be disregarded.

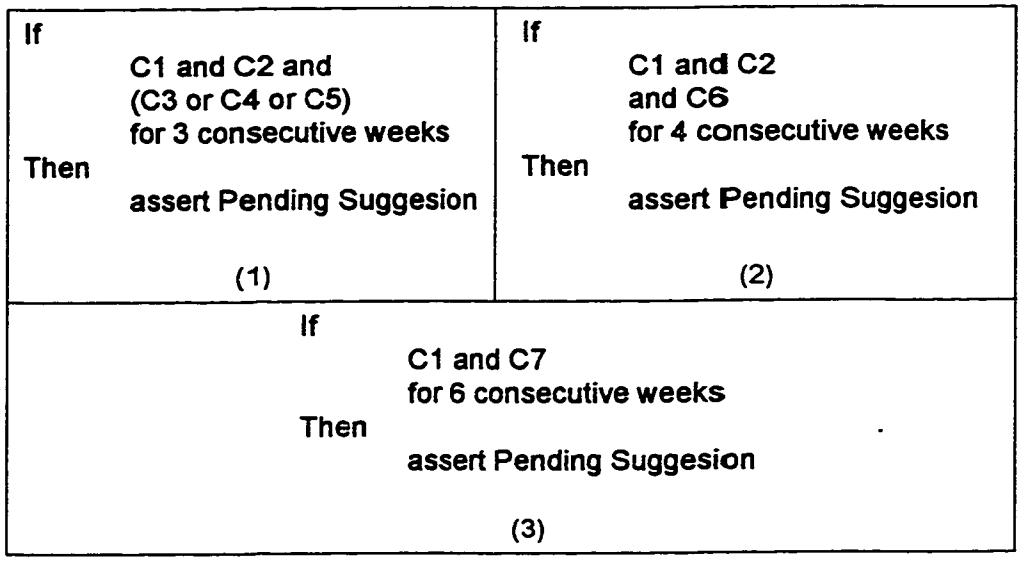

Figure 3.13 Conditions for Pending Suggestions.

The offering suggestions rule set totally depends on the pending suggestions determined by FATMA. Before offering a suggestion, FATMA will scan the pending suggestions table to see which pending suggestions are similar. Two pending suggestions are considered similar if they have the same disposition for the same group of callers or same individual for the same starting and ending times, and the days of one pending suggestion are less and included in the second pending suggestion. When FATMA finds similar pending suggestions, it disregards the pending suggestions that have less days to be suggested, and keep the one with the most days ready to be offered. For example if FATMA finds two pending suggestions:

(pending suggestion: Known, Voice Mail, 9:00-17:00, mon, tue, wed),

(pending suggestion: Known, Voice Mail, 9:00-17:00, mon, tue, wed, thu, fri),

FATMA will disregard the first one and suggest the second one, based on the assumption that such suggestion is best considered to describe the behavior of the user. The pending suggestion disregarded will stay in the pending suggestion table but will not be offered to

the user. When the pending suggestion that was chosen from its similar ones is offered and accepted by the user, other pending suggestions similar to the accepted one will be removed. If the suggestion offered was rejected, the similar pending suggestions would then be reconsidered to be offered to the user.

In my study, I also considered re-offering a suggestion if it was rejected. Since FATMA is assumed to be using a speech user interface, it is more difficult than with a GUI to know the reason why the user rejected the suggestion because the user cannot give the reason for his or her rejection. In this context, I took into consideration two cases for when suggestions are rejected. The first case is that all the suggestions offered at the same time were rejected. FATMA will not know if the user rejected them because they are all wrong, or because the user was too busy to worry about the suggestions at the time the suggestions were offered, or because the user did not know what he or she should do. In this case, FATMA will wait exactly one week to re-offer the same suggestions. While waiting, FATMA keeps observing the user's behavior, and if any change occurred in the behavior of the user regarding any of the suggestions to be re-offered, the suggestion which the user's behavior changed, would be disregarded. The second case, is if at least one of the suggestions offered is accepted. I assumed that FATMA will know that the other suggestions were refused because the user does not want such automation of features. In that case, FATMA will not re-offer the same suggestion unless it observes some change in the behavior of the user. In any of the above cases, we stipulate that a suggestion cannot be offered more than twice. If the user rejects the same suggestion twice, then one of the similar pending suggestions could be considered after another set of

similar observation. If a suggestion is accepted by the user, all similar pending suggestions should be removed from the pending suggestion table.

As described above, the rules will determine what suggestions should be offered, and if they should be re-offered, but the rules do not determine when these suggestions should be offered. When to offer a suggestion is very delicate particularly in the context of a SUI, because when FATMA offers a suggestion, it takes control of the user interface and waits for an answer from the user. Before offering a suggestion FATMA must be sure that the user, is not involved in any other activity, and totally free to be able to understand the suggestion and reply appropriately. In this context, the algorithm I used for FATMA to offer a suggestion is event-based. The suggestion should be relevant with the context of what the user is trying to do. For example, FATMA should not offer a "Call Screening" suggestion if the user is trying to call someone. A "Call Screening" suggestion should be offered only if the user is performing a "Call Screening" feature. The suggestion should be made very timely. For example, FATMA should not offer a "Call Screening" suggestion for the afternoon if the user is performing a "Call Screening" feature in the morning, and FATMA should not offer a "Call Screening" suggestion for a different day unless one of the days in the suggestion matches the day the user is performing the "Call Screening" feature. Furthermore, considering FATMA as a SUI, FATMA should offer a suggestion as soon as the user requests the appropriate feature because if the suggestion is offered in the middle of a process, the user might get lost and not respond appropriately. Moreover if the suggestion is offered at the end of the process, the user might be switching to another feature or quitting FATMA, and might not hear the suggestion completely.

3.4 Test and Demonstration

To demonstrate how FATMA works, I developed a scenario (table 3.1). The scenario is about a family of two working people who set all their calls to Voice Mail in the morning before going to work, They set "known callers" to "Directly to User" at around 5:00pm, and after supper they set all callers back to Directly to User. Not wanting to be disturbed before going to sleep, they first set "Unknown Callers" to "Voice Mail" at 9:00pm and all callers to "Voice Mail" at 11:00pm. On Thursday and Friday, there is a slight change assuming that they go shopping or leave the house at around 7:00pm, they set all callers to "Voice Mail". On Saturdays and Sundays, the dispositions are totally different. Table 3.1 describes in details the dispositions set for a certain week.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
7:00am	All/VM	All/VM	All/VM	All/VM	All/VM	All/VM	All/VM
9:00am						All/DU	All/DU
1:00pm						All/VM	
5:00pm	KC/DU	KC/DU	KC/DU	KC/DU	KC/DU	All/DU	
7:00pm	All/DU	All/DU	All/DU	All/VM	All/VM		
9:00pm	UC/VM	UC/VM	UC/VM	KC/DU	KC/DU	UC/VM	UC/VM
11:00pm	All/VM	All/VM	All/VM	All/VM	All/VM	All/VM	All/VM

Table 3.1 Scenario of a family. KC = Known Callers, UC = Unknown Callers, VM = Voice Mail, DU = Directly to User.

For demonstration's purpose, I created two sets of "Screening Envelopes". The first set are Screening Envelopes describing the table for the first three weeks of usage, and the second set are Screening Envelopes describing the table for the fourth week of usage. Then I assumed the next time FATMA will be used on a Monday morning at 7:00am where the family is going to set the disposition for all callers to Voice Mail. I loaded the first set of Screening Envelopes, ran the rules, I came up with a set of pending suggestions. I used the "Simulation Application" and as soon as I requested the "Call Screening" feature, I got two suggestions from FATMA: the first one is for FATMA to

automatically set Known Callers to Voice Mail for Monday to Friday from 7:00 to 17:00; and the second suggestion was for FATMA to automatically set Unknown Caller to Voice Mail for Monday, Tuesday, and Wednesday from 7:00 to 19:00. For both suggestions I responded by "No". Then, I loaded the second set of Screening Envelopes which describes only the fourth week of usage, I ran the rules, I saw new suggestions determined by FATMA, but I only got the same two suggestions that I had before.

After the first demonstration, I reset FATMA and reloaded the same Screening Envelopes sets. This time, I responded "Yes" to the first suggestion, and "No" to the second suggestion. After I ran the second set and went to the Call Screening Window, this time I only saw new pending suggestions, FATMA had no suggestions to offer.

The third test that I did was to modify the second set of Screening Envelopes by adding to it a fifth week of usage, i.e. the second set had now two weeks worth of Screening Envelopes. I loaded the first set, ran the rules and went to Call Screening Window where I got the same suggestions mentioned above. I accepted the first one and rejected the second one. Then I loaded the second set of Screening Envelopes, and ran the rules. At this point I saw new pending suggestions that were determined by FATMA. Then I went to the Call Screening window, and FATMA offered me the suggestion that I refused before.

With these demonstrations, I tested FATMA to see if it was performing correctly in terms of accepting the commands given to it by the user, asserting the proper envelopes and so on. I was also able to test how FATMA could offer suggestions with the rule sets that I have developed. This concluded that FATMA could be used by any researcher to study the proposed solutions on how to offer "appropriate" suggestions just by designing

and developing sets of new rules, embedding them in FATMA, and analyzing the user reactions.

4 Conclusion

A software system called FATMA (Futuristic Automated Telecommunication Management Agent) is designed and implemented. This is a software tool used for studying various solutions for the problem of "how to provide "appropriate" proactive suggestions related to call processing features utilized by telephone users". Proactive suggestions are considered appropriate when the following criteria are met:

- The suggestions made should be consistent with the previous actions of the user.
- The suggestions should be relevant to the context of what the user is trying to do.
- The suggestions should be timely-based.

FATMA was developed so that user interface designers can study different strategies on how to assist the user by suggesting the use of appropriate call management features in telecommunication. In order to best serve its purpose, FATMA was designed and developed in three different and independent modules: the user interface to FATMA, the event handler, and the rule sets. This modular organization gives researchers the opportunity to change, modify, and add features and rules as they find necessary in an easy fashion without causing any major changes in the other modules.

FATMA was developed so that it can be used in two different ways. The two usages that FATMA offers (the simulation and demonstration applications) give researchers the opportunity to simulate an agent's competency using FATMA and demonstrate how it can offer suggestions.

With the simulation application, researchers study how an agent observes and reacts to the user's actions when the user's actions are consistent and repetitive. The suggestions offered by FATMA as per the rules developed by researchers may be accurate or not, depending on the rules. Although FATMA when used as an agent, may take a few weeks to learn the user's habit and start offering suggestions, the simulation application is crucial for researchers to conduct efficient studies for offering appropriate suggestions.

With the demonstration application, researchers can demonstrate how an agent can observe the users behavior, and how an agent can develop proactive suggestions to be offered to users. Demonstrations are useful in the very early stages of product planning.

4.1 Future Work

As of now, FATMA can offer independent suggestions with respect to a selected feature. If the user wants to set many different features at one time, FATMA will wait for the user to request a specific feature for it to offer the suggestions related to that feature. One way to enhance FATMA, is to enable it to offer multiple suggestions when different features are in use. These suggestions could be sequenced according to suitable criteria.

As mentioned in section 4.3, FATMA can offer a suggestion for a second time, if it is rejected the first time. Re-offering a suggestion sometimes could be annoying to the user. One way to solve this problem and enhance FATMA is by enabling the user to either edit the suggestion to fine tune the feature by changing the time and/or days for that feature, or postpone giving an answer for the suggestion for a later time, or both.

Another way to enhance FATMA is by extending its proactive functionality to provide a level of confidence. The level of confidence will be used by FATMA to

determine if it is ready to start offering suggestions to the user or not. The level of confidence could possibly be based on two thresholds that could be set by the end-user: the *ask me* threshold, and the *do it* threshold[4]. With these two thresholds, three levels of confidence could be developed:

- The level below the *ask me* threshold: FATMA only observes the actions taken by the user.
- The level between the *ask me* and *do it* thresholds: FATMA will offer a suggestion to the user, and will act on behalf of the user only if it is permitted to do so.
- The level above the *do it* threshold: FATMA will act on behalf of the user on a certain feature without offering the suggestion.

Extending FATMA with the level of confidence, will enhance its capability to offer suggestions, not only by observing regularity in the user's behavior, but also by taking into consideration the user's acceptance of FATMA to act on his or her behalf.

References

[1] Wildfire Manual, Emit Corporation, 12 Baby Point Road, Toronto, Ontario, M6S 2E6, Tel: 1-800-wildfire. Http://www.wildfire.com/thanks.html.

[2] Kaye, A.R. and Karam, G., Cooperating Knowledge-Based Assistants for the Office, in: ACM Transactions On Office Information Systems 5, 4 pp.297-326 October 1987.

[3] Lai, K., Malone, T., and Yu, K., Object Lens: A "Spreadsheet" for Cooperative Work, in: ACM Transactions on Office Information Systems, 6, 4 pp.332-353 October 1988.

[4] Kozierok, R., A Learning Approach to Knowledge Acquisition for Intelligent Interface Agents, M.I.T. Media Laboratory, Learning and Common Sense Section, Technical Report. 1993

[5] Stanfill, C. and Waltz, D., Toward Memory-Based Reasoning, in: Communications of the ACM 29, 12 pp.1213-1228 December 1986.

[6] Gilbert, Don, Aparcio, M., Atkinson, B., and Brady, S., The Role of Intelligent Agents in the Information Infrastructure, IBM Corporation, Research Triangle Park, NC USA. 1995. Http://activist.gpl.ibm.com:81/WhitePaper/ptc2.htm.

[7] Lieberman, H., Attaching Interface Agent Software to Applications, Media Laboratory, MIT. 1993. Http://lcs.www.media.mit.edu/groups/agents/papers.html.

[8] Magedanz, T., Rothermel, K., Krause, S., Intelligent Agents: An Emerging Technology for Next Generation Telecommunication?. INFOCOM March 1996.

[9] Caglayan, A., Snorrason, M., Jacoby, J., Mazzu, J., and Jones, R., Lessons from Open Sesame!, A User Interface Agent. Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96). London, pp.61-73. April 1996.

[10] Ali-Ahmad, W., and Velasquez, J., E!Agent: A Friendly E-mail Agent. Center for educational Computing initiatives, MIT. 1995. Http://beirut.mit.edu/MAS962/e_agent.htm.

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

[12] Lashkari, Y., Metral, M., Maes, P., Collaborative Interface Agents. Proceedings of AAAI '94 Conference, Seattle, Washington, pp.444-449. August 1994.

[13] Karmouch, A., Proposal for A Mobile Agent-Based Architecture for Wide Area Networks. University of Ottawa. January 1996.

[14] Grossner, C., Models and Tools for Cooperating Rule-Based Systems. Doctoral Thesis, McGill University, 1994.

[15] Isaacson, P., Wildfire Launches a New Generation of Telephone Voice Assistants and Agents. Http://www.dreamit.com/articles/wildfire.htm. 1995.

[16] Kautz, H., Selman, B., Coen, M., Ketchpel, S., Ramming, C., An Experiment in the Design of Software Agents. Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1. Seattle, WA, USA, July 31 August 4 pp.438-443. 1994.

[17] Murugesan, S., Radhakrishnan T., Intelligent Agents in Telecommunications Applications. Technical Report, Dept. of Computer Science, Concordia University. December 1996.

[18] Wildfire Home Page. Http://www.wildfire.com.

[19] Yankelovich, N., Talking vs. Taking: Speech Access to Remote Computers, ACM CHI '94 Conference Companion, Boston, MA, April 1994.

[20] Yankelovich, N., Levow, G.A., Marx, M. Designing SpeechActs: Issues in Speech User Interfaces. ACM CHI '95 Conference on Human Factors in Computing Systems, Denver, CO, May 1995.

[21] SpeechActs home Page. Http://www.smli.com/research/speech.

[22] Foner, L. N., Maes, P., Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning. The Third International conference on the Simulation of Adaptive Behavior, Brighton, England 1994.