Introduction

The word "**agent**" is often used to describe people who have a helping or assistive relationship to another people. The job of such an agent is to act autonomously to satisfy goals that we may have, give us a greater sense of productivity and reduce our work load. The word agent is used to describe software that plays a similar role-providing help, advice, and "running errands" for the user[5].

In this work, the intelligent agent application we develop is a personal assistant designed to aid us in managing our personal computer.

Two intelligent agents are constructing using Java AWT framework. We denoted to this application by (**PCManager**)which is refer to personal computer manager application, this application provides a graphical user interface using the Java AWT framework[2], and provides the interface to TimerAgent and FileAgent to help us manage the resources in out personal computer

The Agent

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors, a human agent has eyes, ears and other organs for sensors, and hands, legs, mouth, and other body parts for effectors [12].

The ordinary programs in a real world environment sense the world via its input and act on it via output, but is not an agent because its output would not normally effect what it senses later [3].

The agent must have special characteristic. Like adaptability, mobility, transparency and accountability , ruggedness ,self-starters, user centered, autonomy, social ability, reactivity, or proactivity, the agent can operate on the local data and on the internet [12,10].

Artificial Intelligent

Much of the Artificial intelligent community has been working on solving the difficult problem of machine vision and speech, natural language understanding and translation, commonsense reasoning, and robot control. A branch of Artificial intelligent known as connectionism regained popularity and expanded the range f commercial applications through the use of neural networks for data mining, modeling, and adaptive control. Biological methods such as genetic algorithms and alternative logic systems such as fuzzy logic have combined to reenergize the field of Artificial intelligent. Recently, the explosive growth in the Internet and distributed computing has led to the idea of agents that move through the network, interacting with each other and performing tasks for their users. Intelligent agents use the latest Artificial intelligent techniques to provide autonomous, intelligent, and mobile software agents, thereby extending the reach of users across networks[9].

From Artificial Intelligent to Intelligent Agent

The major topics in Artificial intelligent ,search techniques, knowledge representations, reasoning algorithms, and learning can be used by the intelligent agents to enhance the capabilities of applications and help users to get their work done [11].

The Intelligence

An intelligent agent act rationally, It does the things we would do, but not necessarily the same way we would do them. But our agents will perform useful tasks for us. They will make us more productive. They will allow us to do more work in less time, and see more interesting information and less useless data[7]. Our programs will be qualitatively better using AI techniques than they would de otherwise . A software agent exhibits intelligence if it acts rationally .It uses knowledge, information, and reasoning to take reasonable actions in pursuit of a goal or goals[9].

The Perception

In order for a software agent to take some intelligent action, it first has to be able to perceive what is going on around it, to have some idea of the state of the world .for animals, this problem is solved by senses of touch, smell, taste, hearing, and sight. The next problem is to not get overwhelmed by the constant stream of information. Because of the large amount of raw sensory input. The intelligent agent must have an equivalent source of information about the world in which it lives. This information comes in through its *sensors*, which may or may not be grounded in the physical world, the intelligent agent dose not need to have senses in the same way we do, but it still has to be able to gather information about its environment. This could be done actively, by sending message to other agents or systems, or it could be done passively by receiving a stream of event messages from the system, the user or the

other agent. The software agent must be able to distinguish the events (mouse movement)from the significant events (double-click icon).As in windows ,the user generates a constant stream of events to the underling windowing system. The agents can monitor this stream and must recognize sequence of basic user actions[9].

The Action

Once the agent has perceptively recognize that a significant event has occurred, the next step is to take some action. This action could be realize that there is no action to action to take, or it could be to send a message to another agent to take an action on our behalf. like people, agents take actions through effectors. For people, an effectors is our muscles, when we take physical action in the world. Or we can take action through speech (using muscles)or by sending e-mail (different muscles again).By communication with other people, we can cause them to act and change the environment[9].

Intelligent Agent Framework

We develop an intelligent agent architecture using object –oriented techniques .We start with a generic set of requirements and refine them into a set of specifications. We explore how intelligent agents can be used to expand the capabilities of traditional applications and how they can serve as the controller for a group of applications.

Requirements

The first step in any software development project is the collection of requirements from the intended user community. We develop intelligent agents using Java, and provide the ability to add intelligence to applications or applets written in Java[2].An additional fundamental requirement in using the Artificial intelligent.

Functional Specifications

1- It must be easy to add an intelligent agent to an existing Java application.

2- A graphical construction tool must be available to compose agents out of other Java components and other agents.

3- The agents must support relatively sophisticated event-processing capability. Our agent will need to handle event from the outside world other agents, and signal events to outside application[4].

4- domain knowledge must be add to the agent using if-then rules, and support forward and backward rule-based processing with sensors and effectors.

5- The agents must be able to do classification, clustering, and prediction using learning algorithms[5].

6- Multiagent applications must be supported using a KQML-like message protocol[1].

7- The agent should be persistent. That is ,once an agent is constructed there must be a way to save it in a file and reload its state at a later time[8].

PCManager Application

The PCManager allows a user to specify two major functions, Alarms and Watches. Alarms use the Timer Agent to manage the time –keeping aspects of the function. An alarm is a time-based event that can occur at a single specific time or at repeating intervals. For example, we could set an alarm to go off at 6:00 A.M. tomorrow morning, or we could specify an alarm to go off every 10 minutes.

Watches use the capabilities of a File Agent to detect changes to the state of a file or directory in the personal computer file system. A watch could be set to signal when a file is changed or deleted, or when it grows over a specified size

In figure (1) shows the main panel of the PCManeger application. The application interface consists of two parts. The top **awt.list** box

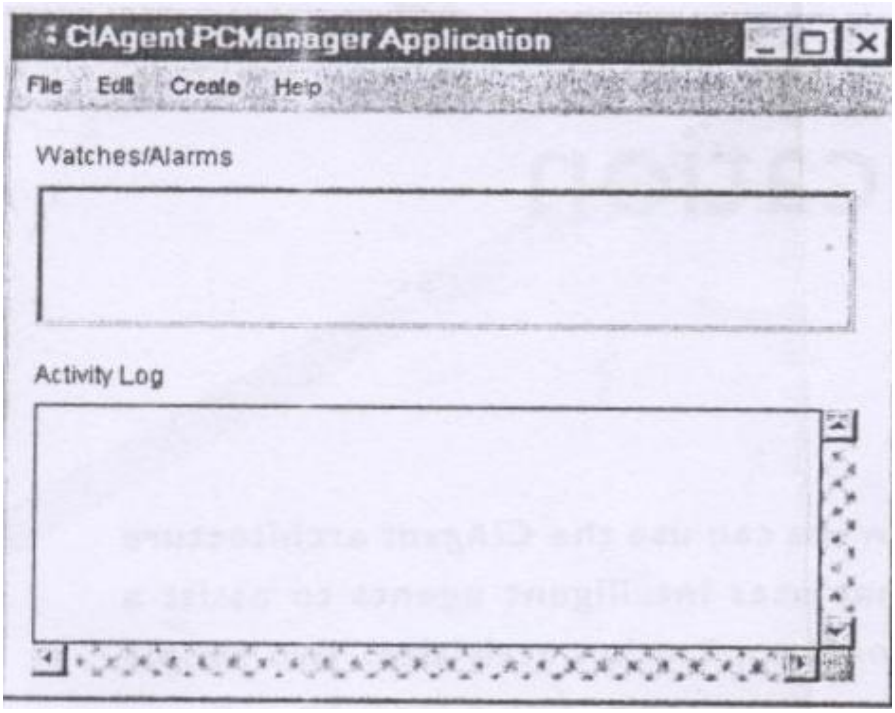

Figure (1) The PCManager application

displays the currently specified Alarms and Watches. The bottom **awt. TextArea** box is used to display status and trace messages.

There are two major secondary panels or dialogs used in the **PCManager** application, the **AlermDialoge** and the **WatchDialog**, which are used to specify the parameters required for Alarm and Watches respectively.

Figure (2) shows the Alarm Dialog. Each alarm has a name, a type, either one-shot alarm or interval and an associated action .A one-shot alarm must have the time of the alarms specified in hours and minutes. The interval alarm requires the specification of the number of seconds between alarms. The user can select from one of three actions when the alarm fires. It can alert the user via an **AlertDialog**.It can

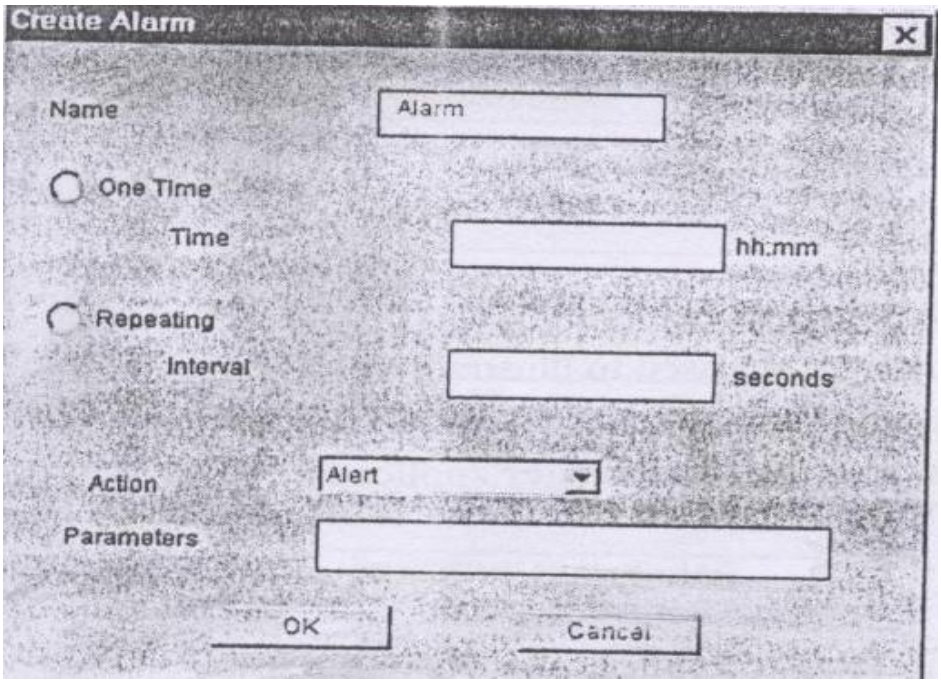

Figure (2) The Alarm Dialog panel.

execute a system command or run application program on the user's behalf.Or,it can send a **AgentEvent** to another Agent. When the user click OK, an instance of a **TimerAgent** is created and its methods are called to set the interval or date, trigger condition, and action. The Alarm name is used as the name of the **TimerAgent** instance.

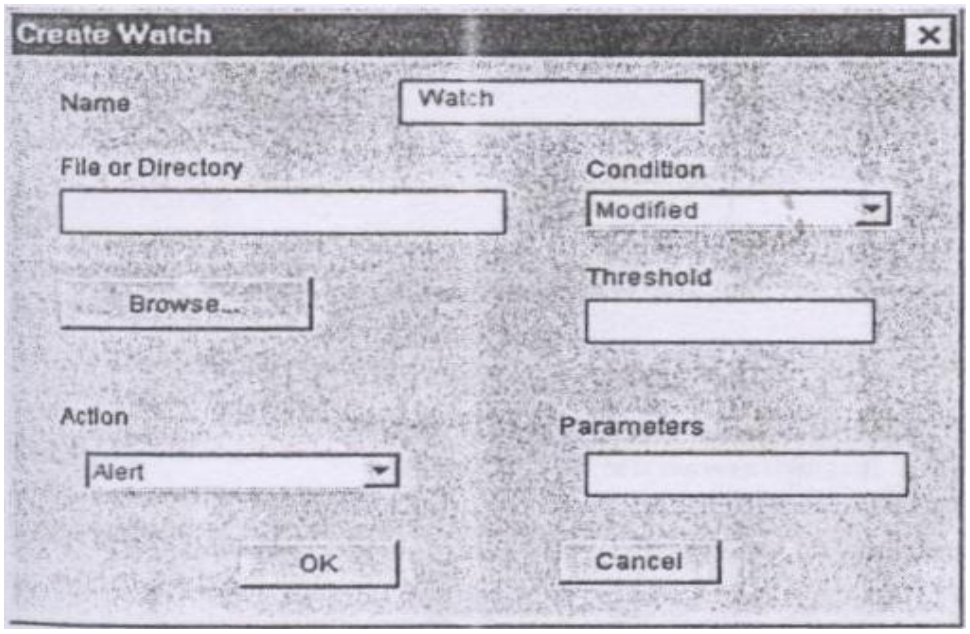

Figure (3) The Watch dialog panel

Figure (3) illustrates the **WatchDialog**. Like an Alarm, a Watch has a name , a parameter to specify the file or directory to be watched ,the trigger condition on the file ,and the action to take when the trigger occurs. If the user does not know the filename or directory path, can click on the **Browse** pushbutton and open a standard **awt, FileDialog** on the PC file system. The trigger conditions include if the file is deleted or modified, or if its size exceeds a specified threshold. Watches support the same three actions as Alarms, alerts, executing commands, and signaling events. When the user click **OK** on the **Watchdialog,** a **FileAgent** is instantiated and the user parameters passed to the instance. The FileAgent's name is set to the Watch name.

Two additional dialogs associated with Alarm and Watch actions are used in this application . The AlertDialog displays an Alert message to the user. The **ExecueDialog** shows the results of an Execute action. Both action dialogs allow the user either to acknowledge them by clicking **OK**, or to cancel the Alarm or Watch with **Cancel** button.

The basic functions provided by Alarms and Watches can be combined to produce interesting behavior. The Alarms will be used to copy one of two files into a target file. This target file will be the focus of a Watch. When all three agents are active in the PCManager application, the scenario is as follows:

1. Alarm 1 copies the file ***test1.dat*** to ***test.dat***.
2. Watch 1 detects that the file ***test.dat*** was modified and signals an alert to user.
3. Alarm 2 copies the file ***test2.dat*** to ***test.dat*.**
4. Watch detects this change to the file, and signals another alert to the user.
5. This sequence repeats indefinitely.

The following figures show the parameter dialogs and the Alert and Execute dialogs generated by this example.

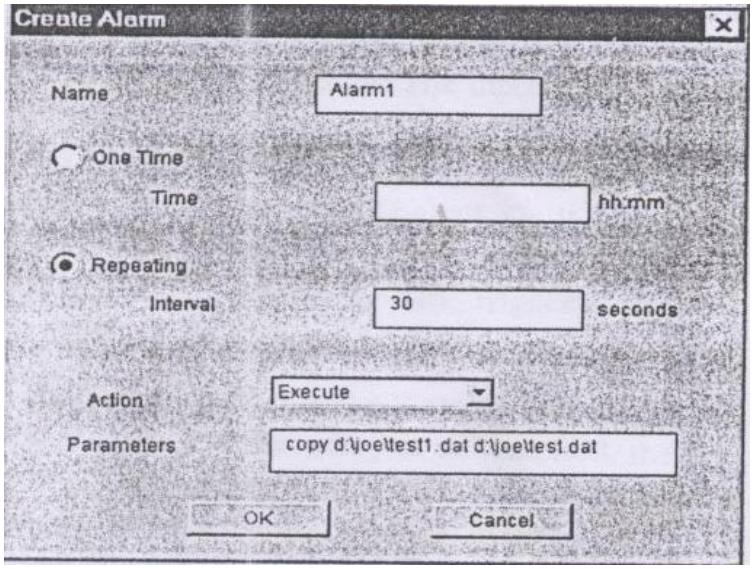

Figure (4) Alarm TimerAgent parameters

Figure (4) and (5) show the parameters used to set up the TimerAgent for Alarm 1 and Alarm 2 respectively. Figure 6 shows the Watch1 setting.

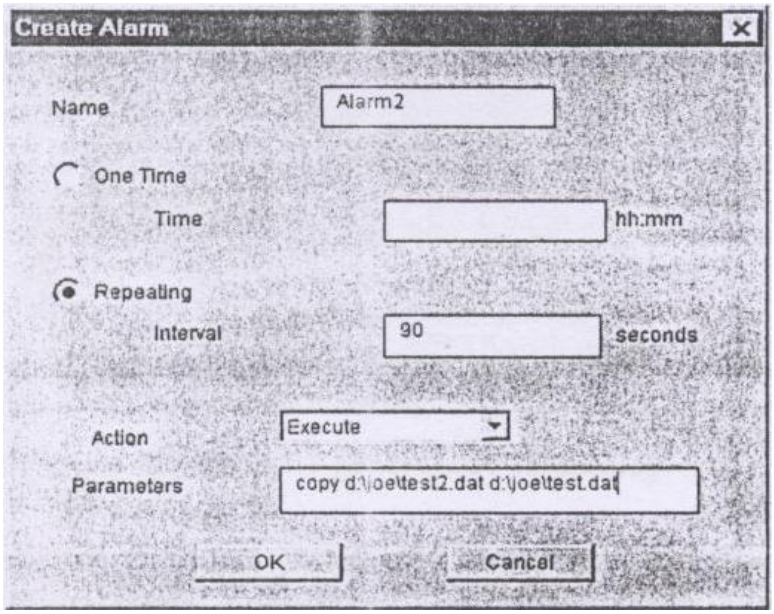

Figure (5) Alarm2 Timer parameters.

When the three agents are created they immediately start running.

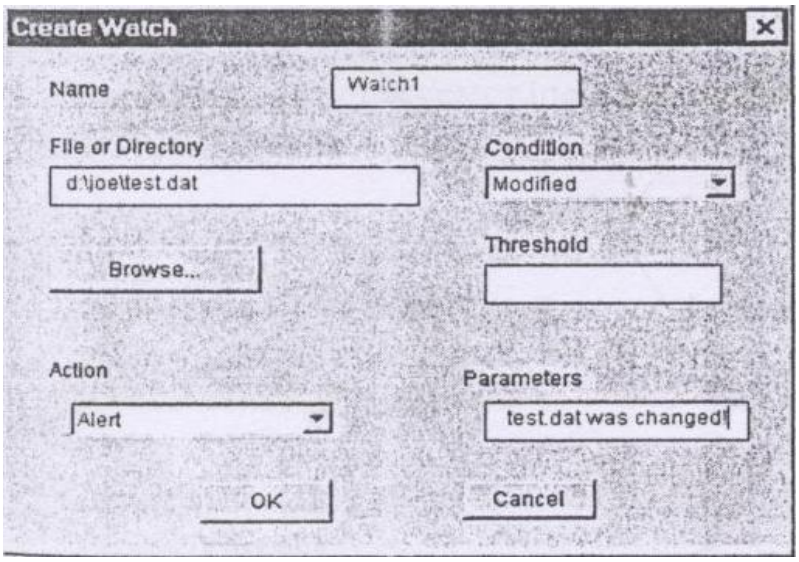

Figure (6) Watch1 fileAgent parameters

Figure (7) shows the PCManager main panel when all of the agents are configured and running. As Alarm1 goes off, the TimerAgent executes the specified parameter string which copies the file *test1.dat* over *test.dat* (Figure (8)).The Watch1 agent wakes up every 15 seconds to check out the status of the *test.dat* file. When it sees that the file has

been modified it signals an Alert. This **AlertDialog** is shown in Figure(9)

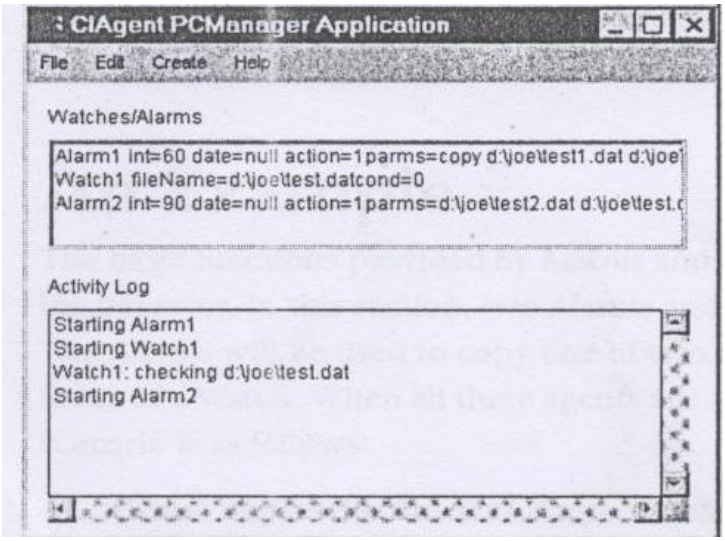

Figure (7) PCManager application example

Alarms(The TimerAgent)

The TimerAgent, is not very intelligent but is certainly autonomous and provides an extremely useful function. A TimerAgent can be used to set one-shout alarms for any specified time, or it can be used to fire recurrent alarms specified intervals. when an alarm condition occurs, the **TimerAgent** can take one of three actions. It can display an Alert, where a dialog pops up on the screen to inform the user that a Timer alarm has gone off. It can execute an arbitrary system command or invoke an application program on the system, passing parameters as required. The TimeAgent can also simply fire AgentEvent to signal another Agent or other Java object.

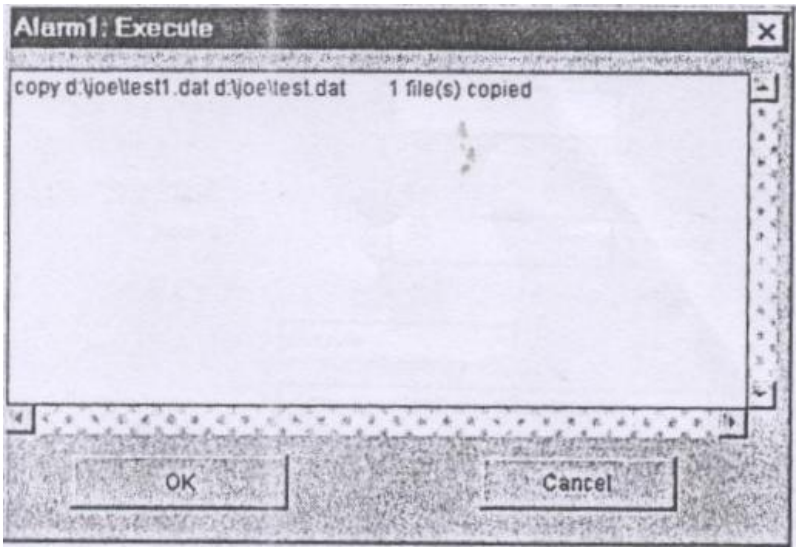

Figure (8) Alarm1 generated ExecuteDialog.

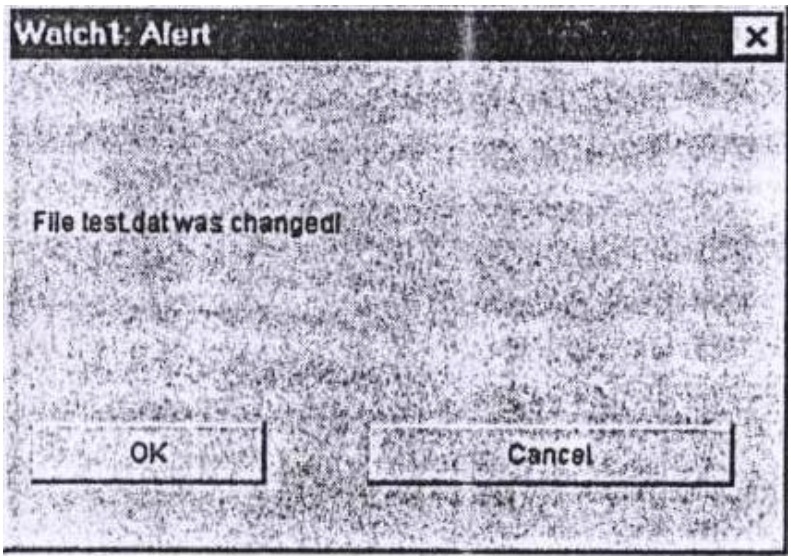

Figure (9) Watch1 generated AlertDialog.

Watches (The FileAgent)

The next Agent is named FileAgent because its purpose in life is to watch a file system and to let the application know when some specified event occurs. The agent can alert the user with the AlertDialog whenever a file is modified, or if the size of a file gets too large (monitoring the size of a swap file, for example). It can also alert the application or another agent whenever the target file is deleted. Like the TimerAgent, the

Fileagent can perform one of three actions when it deleted the watch condition. It can display an alert, execute a command or start an application, or notify another agent of the condition.

Much of the logic of the **FileAgent** is similar to the **TimerAgent**, three watch conditions are supported: if the file is MODIFIED, if the file is DELETED (it dose not in the file system),and if the file exceeds a threshold size in bytes. The name of the file or directory is stored in the *file-Name* **String** member, and information about when it was changed is stored in *lastbanged.*

Conclusions

Special type of software application is presented in this work, which is an intelligent agent, it has become a very popular paradigm in recent years, some of the reasons for this popularity is its flexibility, modularity and general applicability to a wide rang of problems.

The PCManager application allows a user to perform two major functions: set Alarms to go off at specified times or intervals and set Watches on files or directories.

This application represents an example of the utility of autonomous.

References

1. **Finin,T.,Labrou,Y.,and Mayfild, J.(1995).KQML as an agent communication language. In software Agents, J .Bradshaw, ed. Cambridge.**

2. **Flanagan, D.(1996).Java in a Nutsbell. Sebastopol, CA:Ỏ Reilly & Associates.**

3. **Franklin, S & Grasser,"Is it an Agent, or Just a Program?" : A Taxonomy for Autonomous Agents.In Proceedings of the Third International Workshop on Agent, Theories,Architectures,and Languages ,Springer-Verlag,A.(1996).**

4. **Hui-Min Chen," Design and implementation of the Agent BasedEVMSystem",WWW.sims.berkeley.edu/research/metadata /demo.html.**

5. **J.P.Bigus, "Data Mining with Neural Networks" –solving business problems –from Application Development to Decision support ,MG Graw-hill,1996.**

6. **Henry Lieberman and Ted Selker, "Agents for the user Interface ,"Media Laboratory ,Massachusetts Institute of Technology.**

7. **Maes, P.(1991). Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.Cambridge.**

8. **Maes,P. (1994). Agents that reduce work and information overload, Communications of the ACM, 7:31-40.**

9. **McDermott, J. (1982). R1: A rule-based configure of computer systems. Artificial Intelligence,19(1):39-88.**

10. **N.R. Jennings and M. Wooldringe ,"Application of Intelligent Agents", Queen Mary and Westfied College ,University of London.**

11. **Nwana, H.S (1996)). Software agents: An overview.Knowledge Engineering Review,11(3).205-244.**

12. Richard March, Tony Johnson ,"Intelligent software agents", prentice Hall PTR, New Jersey ,1998,http://www.phptr.com.

13. Shirley Lincium, "Introduction to Interface Agent",Lincics@wou.edu,http://www.wou.edu/library/staff/lincicu.com,2003.

14.Silvia Schiafhino, Analia Amandi , "User –Interface Agent Inrteraction": personalization issues, International Journal of Human –Computer studies,60(2004)129-148,www.elsevier.com/locate/ijhcs.