PSERC

Mobile Agent Applications for Power Apparatus Monitoring and Maintenance

Final Project Report

Power Systems Engineering Research Center

A National Science Foundation
Industry/University Cooperative Research Center
since 1996

Power Systems Engineering Research Center

Mobile Agent Applications for Power Apparatus Monitoring and Maintenance

Final Project Report

Power System Monitoring Using Wireless Substation and System-Wide Communications

Part II

Project Team

Mladen Kezunovic, Project Leader
Xiangjun Xu
Texas A&M University

PSERC Publication 02-47

November 2002

Information about this Project

For information about this project contact:

Mladen Kezunovic, Ph.D., P.E.
Eugene E. Webb Professor
Texas A&M University
Department of Electrical Engineering
College Station, TX 77845-3128
Tel. 979-845-7509
Fax. 979-845-9887
Email. kezunov@ee.tamu.edu

Power Systems Engineering Research Center

This is a project report from the Power Systems Engineering Research Center (PSERC). PSERC is a multi-university Center conducting research on challenges facing a restructuring electric power industry and educating the next generation of power engineers. More information about PSERC can be found at the Center's website: http://www.pserc.wisc.edu.

For additional information, contact:

Power Systems Engineering Research Center
Cornell University
428 Phillips Hall
Ithaca, New York 14853
Phone: 607-255-5601
Fax: 607-255-8871

Notice Concerning Copyright Material

Acknowledgements

The work described in this report was sponsored by the Power Systems Engineering Research Center (PSERC). We express our appreciation for the support provided by PSERC's industrial members and by the National Science Foundation under grant NSF EEC-0002917 received under the Industry / University Cooperative Research Center program.

Mitsubishi Electric Research Lab (MERL) provided software and additional financial support for this project. Special thanks are due to Dr. David Wong from MERL, who constantly provided advice for this project. A part of the work has been financially supported by MERL during the summer of 2001 through a summer internship provided for Xiangjun Xu.

Preface

The Power Systems Engineering Research Center sponsored the research project titled "Power System Monitoring Using Wireless and System-Wide Communications." This project consists of two parts:

- Part I. Wireless Communications in Substations
- Part II. Mobile Agent Software Applications.

This is the final report of the second part of the project on "Mobile Agent Software Applications". Part I is contained in a separate report.

Executive Summary

Condition-monitoring data for power apparatus are usually distributed at different locations and reside on heterogeneous computer systems. The diversity of data ownership introduced by power industry restructuring adds more complexity for data sharing and exchanging. A system-wide communication approach is needed to efficiently process and distribute the data.

In this project, mobile agent software has been applied to address the above problems. We have chosen the application scenarios of power apparatus monitoring and maintenance to illustrate certain data processing and exchanging problems.

At the beginning of the project, we studied the functions of the mobile agent technique and compared the technique to other distributed information processing approaches. The most important characteristic of mobile agent software is that the program code, rather than the data, is moved to enable distributed processing over a local area network. This characteristic makes use of the mobile agent technique suitable for situations with distributed data sources. We have illustrated the advantages by implementing three mobile agent software applications.

We have applied mobile agents to solve the problem of scheduling generating unit maintenance**.** In a decentralized operating environment characterizing competitive markets, the historical centralized data processing approach may no longer be feasible. Taking the problem of generating unit maintenance scheduling as an example, an independent system operator (ISO) has network data while a generation company (GENCO) knows only information about its units. They have to coordinate to obtain a feasible maintenance schedule. The proposed approach provides a more secure and flexible way of exchanging data between a GENCO and the ISO.

Circuit breakers play an important role for operational switching and fault isolation. A prototype system has been built to demonstrate the applications of mobile agents in several circuit breaker maintenance scenarios. The system has also been integrated with the circuit breaker monitoring software. Mobile agents travel to the location where the information is located, do the processing locally, and bring back only the results required by the maintenance crew. The underlying distributed information system is transparent to the crew.

Finally, in the third application, we explored the Agent-Oriented approach, which is a further extension of the Object-Oriented approach. Unlike objects that are controlled by external processes, agents usually possess inference abilities and can control their own behavior. This characteristic makes them suitable for modeling active objects. We used agents to model work orders. Each work order has a built-in workflow that is predefined by rules. Once a work order is generated by the maintenance scheduling system, it will automatically continue until the work is finished. No external intervention is required.

Table of Contents

1. Introduction 1
2. Project Background 2
3. Power System Device Monitoring and Maintenance 5
   - 3.1 Device Monitoring 5
   - 3.2 Maintenance Strategies 5
   - 3.3 Maintenance Scheduling in a Decentralized Industry Environment 7
   - 3.4 System-wide Communication and Functional Requirements 7
4. Concordia Mobile Agent Software 9
   - 4.1 The Functions of Concordia 9
   - 4.2 Extending the Functions of Concordia 11
5. Mobile Agent Systems vs. Other Distributed Systems 12
6. Maintenance Scheduling 14
   - 6.1 Scheduling Problems in a Decentralized Industry Environment 14
   - 6.2 Problem Formulation 15
   - 6.3 Solution Method 16
   - 6.4 Implementation 18
     - 6.4.1 Security Consideration 19
     - 6.4.2 Interfacing with Other Systems 21
   - 6.5 Example 23
7. Distributed Information Processing 26
   - 7.1 Heterogeneous Data Access Problems and Requirements 26
   - 7.2 Mobile Agents for Distributed Information Processing 27
     - 7.2.1 Retrieving and Updating Work Orders 28
     - 7.2.2 Retrieving Substation and Circuit Breaker Information 29
     - 7.2.3 Monitoring Circuit Breaker Events 31
     - 7.2.4 Retrieving Information about Spare Parts 31
     - 7.2.5 Preparing Reports 33
8. Integrating with Circuit Breaker Monitoring Software 35
   - 8.1 Circuit Breaker Monitoring Software 35
   - 8.2 Integration 36
9. Agent-Oriented Workflow Management 39
   - 9.1 Function Requirements for Workflow Management 39
   - 9.2 Agent-Oriented Approach 39
   - 9.3 Agent-Based Workflow Management 41
   - 9.4 Further Discussion 44
10. Conclusions and Future Directions 46

References 47

1. Introduction

This document is the final report of the mobile agent part of the PSERC project "Power System Monitoring Using Wireless Substation and System-Wide Communications." To better illustrate system-wide communication problems, several specific application scenarios of power system device monitoring and maintenance have been utilized. Applying the mobile agent technique in those scenarios shows that mobile agent technique is promising for solving system-wide communication problems.

The purpose of this report is to document all the relevant results obtained from the project. The major accomplishments of the project were:

(1) exploring the functions of a mobile agent technique and comparing it to other distributed information processing approaches,

(2) applying mobile agents to solve the maintenance scheduling problem in a decentralized, competitive environment,

(3) demonstrating the applications of mobile agents in several circuit breaker maintenance scenarios,

(4) integrating the mobile agent software with the circuit breaker monitoring software, and

(5) applying the Agent-Oriented approach.

2. Project Background

The "Power System Monitoring Using Wireless Substation and System-Wide Communications" project was aimed at investigating power system monitoring applications from the standpoint of communicating real-time information collected in substations. The first problem that was investigated is the data acquisition and collection in a centralized PC in a substation. An assumption was made that the cost of wiring a dedicated communication network in the substation is prohibitively expensive, resulting in the need to have the connection implemented using wireless communications. Since the option for data processing may range from distributed processing at the point of collection to centralized processing at a PC in the substation, the communication requirements incorporated both options. Once either the raw data or analysis reports are collected in the PC, they need to be further processed and distributed corporate-wide to the maintenance managers and crews. Therefore, the second problem to be studied was approaches for system-wide communications for this purpose. To achieve the required flexibility, the mobile agent communication technology was used to process data contained in various databases located corporate-wide.

Advanced power system monitoring applications rely on both substation and system-wide communications and depend heavily on the flexibility of the communication system. The main issue with substation communications is the wiring cost. The main issue with system-wide communications is the flexibility of allocating the applications in the distributed computer environment. The project has been divided into two parts: the wireless communication part that addressed the first issue and the mobile agent part related to the second one.

This report is about the mobile agent part, which focuses on applying mobile agent software to solve the system-wide communication problems related to power system device monitoring and maintenance. The system-wide communication technology utilized as an example for implementing the distributed processing in the corporate computer network was the Concordia mobile agent technology developed by Mitsubishi Electric Research Lab [1]. This technology enables far more communication and processing flexibility in setting up distributed processing applications and related communications than what is traditionally available in client-server architectures.

The problems and application scenarios we addressed have been depicted in Fig. 2.1. In the scenario, there is an ISO and one power company. Inside the power company (which is surrounded by the dotted-line box) the enterprise maintenance system, the substation computer and the maintenance crew need to communicate with each other. Real-time status monitoring of the devices has been utilized inside the substation. The monitored data is transmitted to the substation computer via wireless communication.

The three agents in Fig. 2.1 represent the three applications studied in this project.

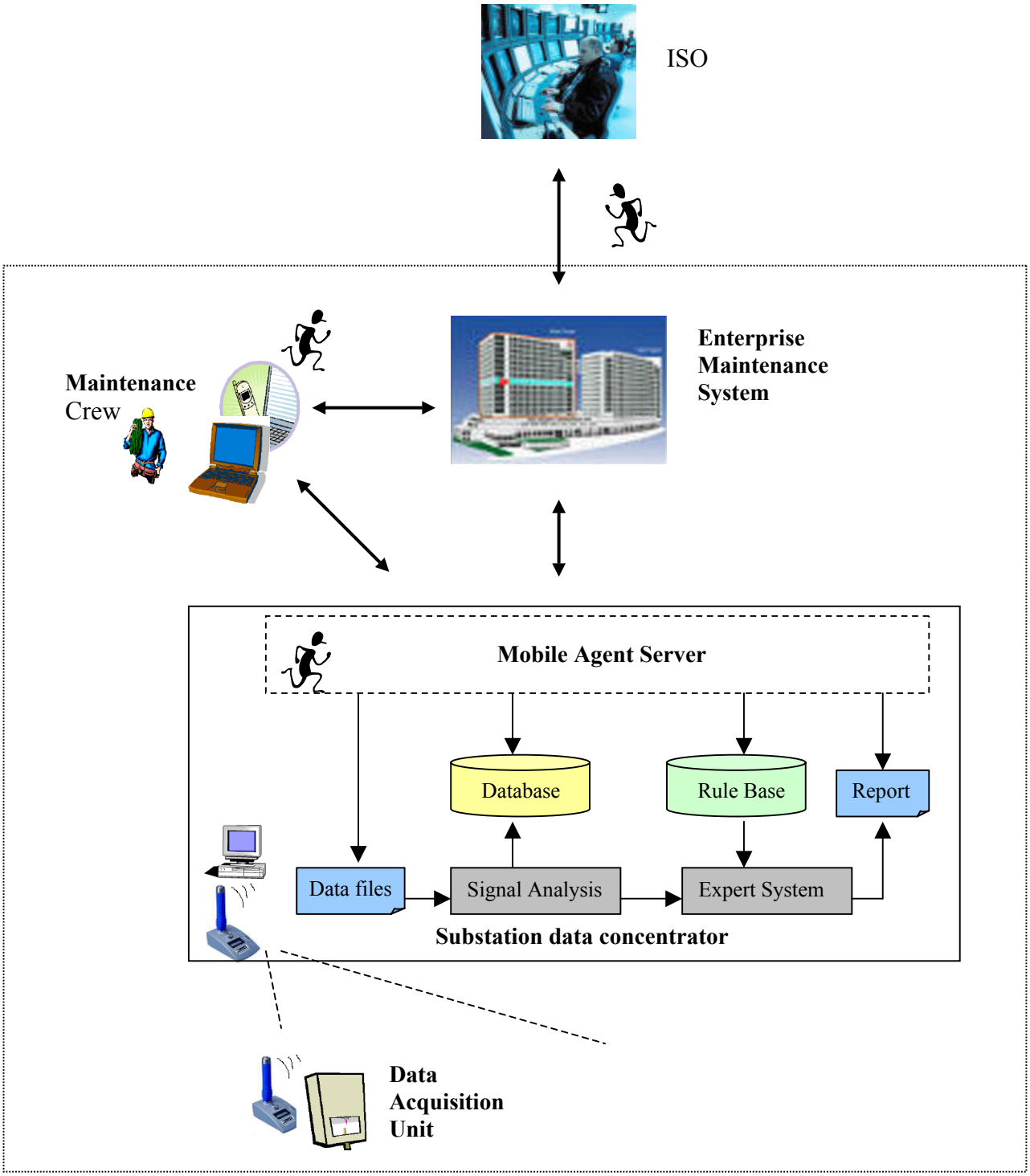

Fig. 2.1 Maintenance system overview

The first application is about maintenance scheduling between the power generating company and the ISO. For maintenance of certain devices that may effect the operation of the whole network, the company might have to get permission from the ISO. Mobile agents have been used

to communicate between the power generating company and the ISO. The results are reported in chapter 6.

The second application is for distributed information processing. The maintenance crew sends out mobile agents to retrieve relevant information and update status. A prototype system has been created. This part of the results is reported in chapter 7.

The third application is to integrate the mobile agent software with the Circuit Breaker Monitoring (CBM) software. The CBM software makes certain real-time information available to both the maintenance system and maintenance crews. Mobile agents have been utilized to distribute those data. This part is reported in chapter 8.

A new Agent-Oriented approach has also been explored. Agent-Oriented approach can provide higher-level abstraction. An example of workflow management has been used to demonstrate its merits in chapter 9.

3. Power System Device Monitoring and Maintenance

3.1 Device Monitoring

With the development of the technology, the number and variety of physical assets of a utility keep increasing. Increasing physical assets imposes a higher burden and more complex requirement on maintenance activities. Some devices are so complicated that only the manufacturer may be able to do the proper maintenance work. Therefore, the traditional maintenance routines need to be updated to reflect the new situation. On the other hand, the devices are becoming increasingly advanced. They may have self-diagnosis and monitoring functions. The continuous condition monitoring of the devices and the real-time data make just-in-time maintenance (or predictive maintenance) possible.

As an example, some new circuit breakers have monitoring instrument available to measure the coil current profiles and operation timing. The recorded information can be transferred to the data concentrators in the substations. With the information about the status of the circuit breakers available, and with the advanced signal processing technique and artificial intelligence applied for feature extraction and pattern recognition, the circuit breakers can be monitored continuously during the normal operation. The circuit breaker status report may be created and disseminated to interested people automatically. In this case, the scheduled test procedure can be eliminated and just-in-time maintenance procedures can be implemented. Both the size of the maintenance crew and the maintenance cost may be reduced demonstrably with this approach.

An on-going project at Texas A&M University (not funded by PSERC) is focusing on online condition assessment of circuit breakers utilizing the monitored waveforms. Online monitoring makes predictive maintenance applicable. The integration of mobile agent software with the Circuit Breaker Monitoring software is discussed in chapter 8.

3.2 Maintenance Strategies

Maintenance plays a vital role in keeping a power system device in a healthy condition. Maintenance philosophies can be classified into three strategies [2, 3].

(1) Run-to-failure (RTF) maintenance strategy, which is corrective in nature. The maintenance is not performed until equipment performance is unacceptable (a failure defined by RCM). RTF has the potential for a great amount of up-time. Equipment is never taken offline for maintenance. RTF allows for a lower skill set because the failures are usually well defined. In addition, there is no requirement to purchase extra parts or labor because there are no maintenance actions between failures. Unfortunately, failures may occur at the most inopportune times, and may severely disrupt production. Repairs are expensive emergencies because there is no resource planning. Parts require express shipment and overtime costs are high.

(2) Time-directed (TD) maintenance strategy, which is preventive in nature. A TD maintenance strategy applies experience and failure history to identify a pattern of degradation, and then attempts to apply specific maintenance actions to return to a desirable level of performance. These actions include lubrication, servicing and overhaul. Correctly applied, TD is successful in extending the interval between failures and maintains high equipment performance. Unfortunately, it also reduces availability by intentionally taking equipment offline to perform maintenance actions. Maintenance cost increases, and more labor and parts are needed. The largest disadvantage is that poor maintenance practice and procedures may create more problems than if the maintenance was never performed.

(3) Condition-directed (CD) maintenance strategy is predictive in nature. A CD maintenance strategy monitors equipment performance to recognize the onset of failure, determine degradation rate and forecast failure. Maintenance actions can be performed at the optimum time before failure. Although CD has nearly none of the disadvantages of the RTF and TD strategies, and is effective at recognizing degradation before failure, it may be very expensive to implement. It also requires a higher level of skills for operators along with more extensive training.

The different maintenance philosophies are summarized in Table 2.1.

Table 2.1 Maintenance Philosophies

Strategy	Description
Corrective maintenance	The repair and restoration of equipment or components that have failed or are malfunctioning, and are not performing their intended function.
Preventive maintenance	The periodic and planned maintenance actions taken to maintain a piece of equipment within the expected operating conditions. It extends the equipment life and is performed prior to equipment failure to prevent equipment failure. Includes technical specification surveillance, in-service inspections, and other regulatory forms of preventive maintenance.
Predictive maintenance	The continuous or periodic monitoring and diagnosis of equipment in order to forecast component degradation so that as-needed planned maintenance can be performed prior to equipment failure. Not all equipment conditions can be monitored and failure modes predicted; therefore, predictive maintenance must be selectively applied. Reliable predictive maintenance is normally preferable to periodic internal inspections or equipment overhauls.

According to a survey by the IEEE Risk and Probability Applications Subcommittee, preventive maintenance performed in fixed time intervals is the most adopted strategy [3].

Each maintenance strategy has its own advantages and disadvantages, and thus most suitable application scenarios. For example, in some cases, the predictive maintenance may improve the

reliability and the economy over the time-directed maintenance strategy. In other cases, run-to-failure maintenance strategy may be most suitable. A systematic solution is to utilize the Reliability Centered Maintenance (RCM) methodology, which performs analysis of failure modes and cause-effect analysis on the devices, and then tries to find which strategy is the most cost-effective and appropriate for an application. Usually, RCM focuses routine maintenance on only those items considered critical for continued reliable operation of important system functions. It also helps to establish appropriate maintenance tasks and intervals based on specific causes of equipment failure and other equipment condition indicators. RCM determines the functionality of the equipment, assesses consequences of potential failure, and applies the most consistent strategy toward those consequences. According to the results given by RCM, a device may be more suitable for condition-directed or time-directed maintenance, or even no maintenance at all.

RCM is still an empirical approach, and requires acquiring extensive amounts of data to operate. A more accurate approach would be to utilize mathematical models (deterministic or probabilistic ones) [3].

3.3 Maintenance Scheduling in a Decentralized Industry Environment

The above mentioned device maintenance involves only activities within the company owning the devices. However, for certain devices whose maintenance would affect the normal network operation, the maintenance schedule needs to be approved by the network operator.

Power system facility maintenance involves long-term scheduling, short-term scheduling, and the actual maintenance work. In traditional maintenance scheduling, all the procedures can be done in one utility. This is no longer true in a decentralized, competitive industry environment, where the situations become more complicated due to the diversities of the ownership of the equipment and the availability of the information. Inevitably, equipment maintenance in the new decentralized environment may involve more coordination and communication among different entities. For example, the system level maintenance scheduling may involve the ISO, TRANSCO and GENCO.

Currently, the GENCOs and TRANSCOs need to submit maintenance schedules to the ISO for approval and the ISO coordinates all the submitted schedules considering system constraints. There may not be a feasible solution at first iteration. Negotiation among the ISO and different GENCOs and TRANSCOs is required. The whole process is time consuming and tedious.

Decomposition techniques, such as Benders decomposition and Dantzig-Wolfe decomposition, can be used to reformulate the whole scheduling problem. In general, the whole problem is decomposed into one master problem and several sub-problems that are solved iteratively until a proper solution is obtained [4].

3.4 System-wide Communication and Functional Requirements

Many distributed data and information sources are involved during the monitoring and maintenance process. Some examples are the monitored real time data at the substation, the spare parts information at the warehouses, the maintenance schedules and status information at the

enterprise center, etc. A system-wide communication system, required to help in accomplishing maintenance tasks more efficiently, should possess the following capabilities:

- Working on heterogeneous systems
- Working on low-bandwidth slow networks and occasionally connected networks
- Supporting user level authentication and access control
- Supporting encrypted data transmission
- Providing friendly user interfaces and programming interfaces.

The first requirement is due to the varieties of computer systems the communication system needs to deal with. Even inside one company, there are most probably multiple types of platforms and operating systems. A typical combination consists of legacy mainframe computers, Unix servers and workstations, and PC's. The mobile devices, such as handheld computers, also add diversity.

Slow and low-bandwidth connections may exist within certain parts of the company. Dial-up access via telephone lines and mobile communication using wireless connections are examples. Those slow communication links are often connected only on an as-need basis.

The security issue is always an important concern. Users should be authenticated and only allowed to access authorized information. In addition, the data should be encrypted during transmission to prevent unauthorized accessing and tampering.

4. Concordia Mobile Agent Software

The term agent is used increasingly to describe a broad range of computational entities. Some agents perform tasks individually; others need to work together. Some agents are mobile, whereas others are static. Some agents communicate via messages, and some agents can learn and adapt. The agents can be divided according to their main functions, such as collaborative agents, smart agents, interface agents, reactive agents and mobile agents [5].

The three primary characteristics of an agent are autonomy, intelligence and mobility. Autonomy means the agent should control its behavior rather than let the user do so. Agents should be able to spontaneously act upon particular environment changes. Furthermore, certain agents can act proactively via prediction and planning. Intelligence is important to keep agents autonomous. By acting like human beings, agents can be perceived as possessing certain intelligence. Mobility is not a necessary requirement for agents, but it can help improve work efficiency in certain cases.

4.1 The Functions of Concordia

The Mitsubishi Electric Research Lab's Concordia mobile agent software is a full-featured framework for development and management of network-efficient mobile agent applications for accessing information anytime, anywhere and on any device supporting Java [1]. An agent application program can travel through the internet/intranet to the computers where the Concordia server or transporter is running. Compared to distributed database systems, an agent can process the data locally and thus reduce the network traffic. The Java platform and Concordia software make the actual communication work transparent to the agents, so the programming tasks have been greatly simplified. Moreover, the data processing work can be done at the most appropriate place, thus the resource utilization can be optimized. Another advantage of the agent architecture is the ability to be easily expanded; e.g., an agent's functionality can be enhanced using artificial intelligence.

In the Concordia mobile agent framework, the device must have the Concordia server (or the lightweight Transporter) running for the mobile agents to be able to travel to the device. The Concordia server can run on any platforms where the Java runtime environment is available. As shown in Fig. 4.1, the devices without JVM (Java Virtual Machine) are supported through a communication node. The communication node can use any proprietary protocols to talk with the mobile devices. As long as the communication node has the Concordia server running and exposes the communication functions to the agent through some programming interface, the agent can communicate with the mobile devices. In this case, since the Java environment cannot cover the whole range of the devices, the mobile agents need to know the programming interface in order to communicate with the mobile devices.

Sun has released the Java™ 2 Platform, Micro Edition (J2ME) and the Mobile Information Device (MID) profile for the Palm OS platform [6]. One can expect that the Java platform will be available on other mobile wireless devices soon. Since mobile device manufacturers will take

care of the communication details, there will be no barrier to where the agents can travel. If all the devices have Java VMs (or KVM) and Concordia servers running, the communication will be thoroughly encapsulated and programming work involved in agent development can be greatly simplified.

The Internet provides a global infrastructure for sharing resources, services and computing power. Mobile agents offer a configurable, scalable and active computing platform over the Internet; the mobile agent paradigm is ideal for distributed Internet computing.

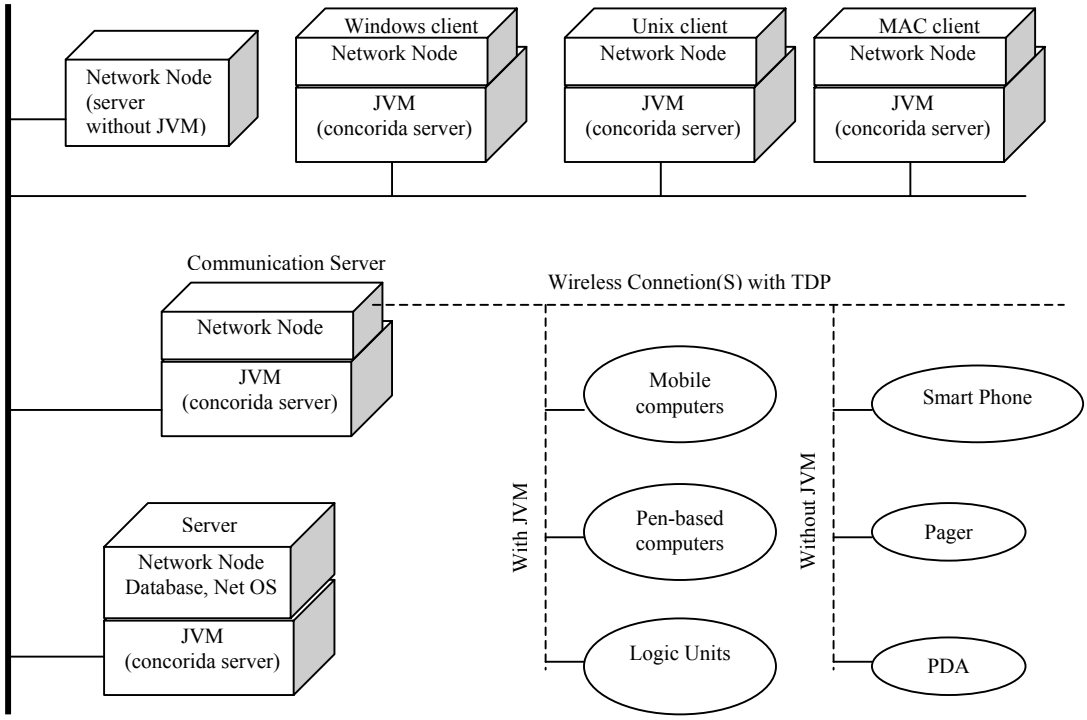

Fig. 4.1 A typical CONCORDIA system setup

For supporting mobile agents, Concordia provides the following functions:

- Processing data even if the user is disconnected from the network
- Accessing and delivering information across multiple networks using wire-line or wireless communication
- Supporting multiple client devices, such as Desktops, notebooks, PDAs, and smart cell phones
- Supporting distributed events (both selected and group events)
- Supporting agent collaborations
- Supporting access to local services via service bridges
- Supporting security features like encrypted transmission and user authentication
- Supporting lightweight agent transporter for GUI applications.

Since the Concordia framework is built on a Java platform, other functions supported by a Java platform are also available for Concordia mobile agents. Among them, the JDBC (Java Database Connectivity) interface to access database and RMI (Remote Method Invocation) interface for distributed objects are the most notable ones [7,8].

4.2 Extending the Functions of Concordia

The Concordia framework provides good support for the mobility of agents. However, it lacks support for other characteristics of software agents, e.g. intelligence.

Jess (Java Expert System Shell) is an expert system shell and a scripting language written entirely in Java language [9]. Jess supports the development of rule-based expert systems. Its rule engine uses an improved form of a well-known algorithm called Rete to match rules against the knowledge base. The Jess language is very similar to the language defined by the CLIPS expert system shell [10]. Jess can be used for knowledge representation, learning and reasoning. By integrating Jess into Concordia software, the agent may act in a more rational and intelligent way.

Artificial Intelligence technologies, such as learning and planning, will have a great impact on the directions of agent technology. The ability to make flexible plans enables mobile agents to make a plan based on inputs received from users and adopt themselves according to information gathered from the network. For example, the Concordia framework is utilizing a user-specifiable itinerary to define the actions of a mobile agent. In this case, the user must know a lot about the situation at each node and plan an optimal routine to achieve a certain task. The approach becomes impractical when the number of nodes and resources goes beyond some limitation. A better way seems to be integrating some intelligence into the mobile agent and the users need only to define ‘what to do’ rather than ‘how to do’. The framework itself will have some mechanism to calculate an itinerary for a specified task.

We have explored the integration between Concordia Ver. 1.1.7 and Jess Ver. 6. The application can be found in chapter 9.

Other functions of interest defined by FIPA (Foundation for Intelligent Physical Agents) include agent communication language, inter-agent collaboration, and ontology [11]. Those functions are necessary to implement a multi-agent system.

5. Mobile Agent Systems vs. Other Distributed Systems

The mobility aspect of software agents mainly focuses on the problems of distributed information processing. The mobile agents can travel to the place where the information is located, perform local processing, and bring back only the final results. In this model, it is the processing code that has been moving around instead of the data to be processed. The rationale behind this is that sometimes the processing code itself is much smaller than the data.

There are several reasons that make moving processing code around (code migration) more attractive in some cases. First, if the size of the data to be processed is very big, then it can usually save network bandwidth by transferring the code. Second, moving the processing from heavily loaded computers to lightly loaded ones can help improve system performance. Third, more flexibility can be achieved. For example, now the processing software can be dynamically updated [12].

Of course, there are several difficulties involved in implementing code migration. The most notable ones are (1) working on heterogeneous systems and (2) ensuring security [12]. Codes may have certain system-specific features that prevent them running directly on a different platform. In addition, the users must be ensured that the codes that travel to their machine perform only the authorized operations. A well-known example is the Java Applet and ActiveX controls embedded in web pages. When those applets and controls are downloaded into the user computer, certain mechanisms must be used to prevent them from doing malicious things. As discussed in the last chapter, Concordia has good support for security problems.

To move a process from one computer system to another, the minimum requirement is to transfer the code segment and some initialization data of the process, which is called *weak mobility*. A transferred program will always start execution from its initial state. In contrast to weak mobility, *strong mobility* (execution migration) requires that a running process can be stopped, moved to another place, after which the execution can be resumed where it left off [12]. Fully implemented transport of agent involves the code segment, the data segment and the execution state. Concordia supports code and data migration, and thus it supports weak mobility.

In distributed information systems, the data stay at their original sources rather than being transferred to a central place. By eliminating the central information source, the problem of data transfer and synchronization has been eliminated. However, some new problems are created. First, the distributed information sources are most probably residing in heterogeneous systems with different formats. The conventional solution is to standardize either the data format or a set of Application Programming Interface (API). XML is an effort to solve the data format problem, whereas CORBA, COM, J2EE represent the efforts of the API approach [8, 13]. The approach of mobile agents is rather different. It does define the standards of API or data format, but makes the processing codes easier to be replaced [1].

With the support of the underlying agent platform (e.g., Concordia), agents with different processing codes can be sent to the appropriate information sources. With the knowledge of the data format and accessing method, an agent will have no problem in processing the information. When new information sources are added, new agents can be made to deal with that specific type of information sources.

Another type of distributed information systems is formed due to the lack of interoperability among subsystems. For example, there are usually several different information subsystems in a certain company. The company may purchase the accounting system from vendor A and the customer relationship management system from vendor B. Generally, no single vendor can meet all the requirements of a certain company; therefore, the company must have different subsystems to fulfill its different function requirements. Those subsystems compose the whole information system of the company. Problems usually exist in integrating the subsystems. Human intervention and paper-based information exchange are usually required. This type of distributed information systems caused by interoperability can be seen as a type of functionally distributed systems, since each subsystem fulfills part of the total functions.

To better integrate the distributed information systems (both geographically and functionally), a higher-level integration approach is desired. The approaches to standardize the data format or API are not very suitable because they expose too many details in each subsystem. The tight coupling relationship among subsystems also makes changes and updates expensive.

Software agent techniques may provide a solution to this problem. First, the agents can communicate with each other using higher-level languages, which is more abstract and thus isolated from the details. Second, the agents can incorporate some intelligence. One application is to integrate different subsystems by mimicking the above-mentioned human intervention behavior. Third, software agents can travel securely through the internet/intranet to the computers where the information is located and process the information locally. Furthermore, the mobile agents are self-upgradable, making it relatively easier to implement applications and have different types of workflow based on the same set of information sources. In this way, a loosely coupling distributed information system can be built.

6. Maintenance Scheduling

6.1 Scheduling Problems in a Decentralized Industry Environment

The power system equipment maintenance involves scheduling, both long-term and short-term, and executing the actual maintenance work. In a traditional maintenance scheduling process, all the steps are usually done in one utility. However, it is no longer true in a decentralized industry environment where the situation becomes more complicated due to the diversities of the ownership of the equipment, and to limited available information. Inevitably, equipment maintenance in the new industry environment involves the need for more coordination and communication among different entities. This leads to the requirement for an information exchange mechanism among market participants for the purpose of maintenance scheduling, and thus resulting in the need for development of new tools for maintenance scheduling.

A typical maintenance scheduling procedure in decentralized environments involves the planning, submitting and approving steps [4]. Initially, an asset owner makes a tentative schedule/plan for the maintenance, and then the initial schedule is submitted to another entity, usually the ISO, for approval. The ISO checks the submitted schedule against the system constraints to decide its feasibility. If it is not feasible, then the entity will be asked to modify its maintenance schedule. The whole process may be tedious and very complicated when more than two parties are involved.

The authors in [4] applied Benders decomposition method to deal with the new maintenance scheduling issues in the decentralized environment. The basic idea is to decouple the whole scheduling problem into one master problem and several sub-problems. The master problem is solved on the asset owner's side, whereas each sub-problem represents a set of constraints imposed by a third entity. An iterative process is usually needed until the solution converges or no solution conclusion is reached. In [4], how the actual iteration proceeds is not discussed. This may be a problem in the real world (since different information exchange formats may exist) if an automated scheduling system is to be built. Exchanging information on paper and involving human beings to create the input and output may be time consuming and error-prone.

The Concordia mobile agent software provides a flexible framework for mobile agent applications [1]. An agent application program can travel through the internet/intranet to the computers where the Concordia server or transporter is running. Concordia also supports Distributed Events, Agent Collaboration and Service Bridge. Commercial versions of Concordia add encryption and proxy support. Compared with distributed database systems, an agent can process the data locally and thus reduce the network traffic. The Java platform encapsulates the network layer from the agent, which makes the programming easier. The Concordia mobile agent may fit very well in the above-mentioned iterative process of making the maintenance schedules.

The agent is a natural concept for representing an entity. By integrating some gaming strategies into an agent, it can represent the client doing the time-consuming negotiation work, especially

when some conflicts need resolving or other types of negotiating are needed. On the other hand, the exact system models are difficult or too complicated to establish in some cases, so the conventional "hard" computing cannot easily be implemented. Intelligent agents with heuristic experience captured from human experts may become very useful.

6.2 Problem Formulation

In general, the maintenance-scheduling problem is determined by the adopted maintenance strategy. The current commonly used maintenance strategy relies on preventive maintenance with fixed time intervals. The basic rational behind this is that "devices should be overhauled regularly". A more systematic solution is to perform Reliability Centered Maintenance (RCM) analysis, and determine the most proper maintenance strategy for each device. The analyses performed in RCM include function and failure definition, failure modes and effects analysis, and finally, the selection of maintenance actions [2].

Even if the practice of periodically performing preventive maintenance is adopted, the beginning time of the maintenance may still change in a given range; therefore, there will be a problem of optimization. The optimization problem can be modeled in two ways [4]. The first one is based on fictitious costs to penalize deviations from a preferred maintenance schedule. The second approach is based on identifying maintenance windows (time intervals) in which the preferred maintenance schedule is represented within windows, and the objective or goal is to minimize the real maintenance cost instead of a fictitious cost. The maintenance window is shown in Fig. 6.1. e_i and l_i represent the beginning and the ending of the window respectively. The maintenance begins at the time of s_i and lasts for a time length d_i. s_i must fall into the interval between e_i and l_i.

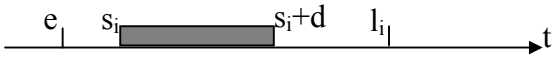

Fig. 6.1 Maintenance Window

The ultimate goal of the optimization is to maximize the income, which is the revenue minus the production and maintenance cost. From the viewpoint of maintenance, the objective can be equalized to minimize the maintenance cost plus the loss of revenue due to the maintenance. The constraints may include maintenance constraints (maintenance policy, maintenance crew availability, etc), and system constraints.

Mathematically, long-term generation maintenance scheduling problem can be formulated as follows (equations 1-4):

$$Min \quad \sum_t \sum_i \{C_{it}(1 - x_{it}) + c_{it} g_{it}\} \tag{1}$$

subject to:

$x_{it} = 1$ for $t \le e_i$ or $t \ge l_i + d_i$ (2)

$x_{it} = 0$ for $s_i \le t \le s_i + d_i$

Maintenance constraints (3)

System constraints (4)

where:

C_{it} - maintenance cost of unit *i* at time *t*

x_{it} - status of unit *i* at time *t*, $x_{it} = 0$ stands for maintenance

c_{it} - lost revenue of unit *i* at time *t* due to maintenance

g_{it} - generation output of unit *i* at time *t*

s_i - maintenance variable of unit *i*, start of maintenance

e_i, l_i - maintenance window for s_i

d_i – the duration of maintenance of unit *i*.

Constraints in (2) represent the maintenance window, which reflects the fixed time-interval preventive maintenance strategy. The maintenance constraints may include the crew and resource availability, seasonal limitations, and other constraints such as fuel and emission constraints. The system constraints represent the peak load balance, transmission flow limits and energy reserve requirements, which are usually not available to a generating company (GENCO) making the schedules.

There are several simplifications in the above formulation. First, the maintenance strategy may differ from the assumed fixed time-interval preventive one, so the maintenance window method is no longer applicable. Second, in a decentralized environment, the price of electricity may not be fixed for an extended period, and the generation companies need to consider the price fluctuation when making their maintenance schedules. The actual maintenance-scheduling problem is more complicated, perhaps making some rule-based heuristic solutions preferable since the mathematical model becomes too complicated to build. Third, there is usually more than one GENCOs submitting maintenance schedules to the ISO in the real world. The ISO must have a strategy to resolve conflicts among GENCOs. The mobile agent architecture and intelligent agent technique may be very suitable for handling all of these factors.

6.3 Solution Method

The problem given in (1) is a Mixed-Integer Programming (MIP) problem. In general, the number of generating units is usually not very big, so the problem generally can be solved efficiently. But, the system constraints are usually not available to a GENCO in a decentralized environment, which means a trial schedule without system constraints must be defined at first and then be submitted to the ISO for approval. The ISO will check the feasibility of the schedule, and either approve it or deny it. When a proposed schedule is denied, the GENCO would prefer if the ISO could feedback some information on how to modify the initial schedule to meet the system constraints.

For the generation unit maintenance problem given in (1), Benders decomposition method can be used to decompose it into a GENCO master problem and an ISO sub-problem. The master problem is first solved without system constraints to get the decision variables. Once those variables are fixed, the ISO sub-problem can be solved with system constraints. If the initial trial schedule is not feasible, a Benders cut is generated. The cut is added into the master problem in the next iteration. The whole process is depicted in Fig. 6.2.

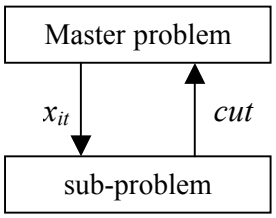

Fig. 6.2 Iteration of solution process

Benders Decomposition: [4]

To make the illustration more clear, the mixed-integer problem of (1) is rewritten as the following form:

$$\begin{aligned} &Min \quad c^T x + f(y) \\ &S.T. \quad Ax + F(y) \geq b \\ &\qquad x \geq 0 \\ &\qquad y \in Z \end{aligned} \tag{5}$$

where:
A: constraint matrix
x: continuous variable vector
y: integer variable vector

If values of y are fixed at first by choosing:

$$y \in R = \{y \mid Ax \geq b - F(y), x \geq 0\} \tag{6}$$

Then (5) is linear in x, and it can be written as:

$$\min\left\{f(y) + \min\{c^T x \mid Ax \geq b - F(y), x \geq 0\}\right\} \tag{7}$$

Based on the duality theory, the set R in (6) may be rewritten as:

$$R = \left\{y \mid (b - F(y))^T u_i{}^r, i = 1 \ldots n_r\right\} \tag{8}$$

where u_i^r is the extreme point vector which belongs to cone C = {u | ATu ≤ 0, u ≥ 0}, and n_r is number of extreme points of cone C. The inner minimization in (7) can be rewritten as follows:

$$\begin{aligned} &Min \quad c^T x \\ &S.T. \quad Ax \geq b - F(y) \\ &\qquad x \geq 0 \end{aligned} \tag{9}$$

Its dual problem is:

$$\begin{aligned} &Max \quad (b - F(y))^T u \\ &S.T. \quad A^T u \leq c \\ &\qquad u \geq 0 \end{aligned} \tag{10}$$

Substituting (10) into (7) yields a new form of (5):

$$\min\left\{f(y) + \max\{(b - F(y))^T u \mid A^T u \leq c, u \geq 0\}\right\} \tag{11}$$

(11) is equivalent to the following procedure:

$$\begin{aligned} &Min \quad z \\ &S.T. \quad z \geq f(y) + (b - F(y))^T u_i^p, \quad i = 1 \ldots n_p \\ &\qquad (b - F(y))^T u_i^r \leq 0, \quad i = 1 \ldots n_r \end{aligned} \tag{12}$$

where u_i^p is an extreme point of P = {u | ATu ≤ c, u ≥ 0}. Problem (12) is equivalent to problem (5) with integer variable *y*. Problem (12) has one constraint for each extreme point that translates into an enormous number of constraints even in a problem with moderate dimensions. However, only a small fraction of constraints will be binding at an optimal solution. Therefore, we begin with a few constraints and solve (12), which is the master problem. Either sub-problem (9) or (10) is used to see if this solution satisfies the remaining constraints.

The solution method mentioned above requires an implementation with the following features.

- It shall work on heterogeneous hardware platforms and operating systems. The iteration process involves different entities, such as the GENCO and the ISO, which may have different hardware and software environments.
- It shall be able to authenticate and authorize the users.
- It shall provide secure communicate channels in the cases that the public network is used to connect the GENCO and the ISO.
- It shall integrate with the existing systems well.

6.4 Implementation

The maintenance scheduling system is implemented using Concordia mobile agent software. Two computers have been setup to represent the GENCO and the ISO respectively, as shown in Fig. 6.3. Both computers have the Concordia server running.

The mobile agent travels between those two servers and calls some local services provided by the servers. The services may be provided by other Java software, or exported from legacy systems using Java Native Interface (JNI) or Common Object Request Broker Architecture (CORBA) [13].

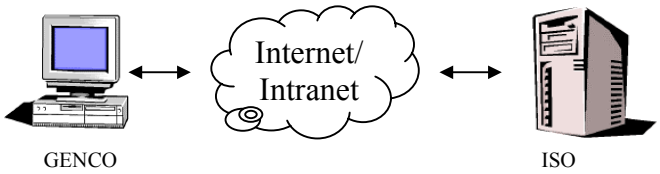

Fig. 6.3 Computer setup

When solving maintenance scheduling problems, the mobile agent architecture provides a number of advantages.

- The mobile agents can be authenticated when they arrive at an entity (GENCO or ISO) to determine their identification.
- An entity can control the information the mobile agents can access according to their identification.
- An entity can expose its services to mobile agents and the entity can control accessibility to the services.
- The software structure becomes more flexible by decoupling the overall negotiation process and the local problem solving processes (at GENCO or ISO). For example, the GENCO may use different methods to generate the maintenance schedule, and may use a different data format for the schedule. As long as the service interfaces are not changed, the whole system structure will not be affected.

6.4.1 Security Consideration

Two apparent security problems arise when applying mobile agents. First, the mobile agents need to be authenticated and authorized at the servers. Second, to ensure the integrity of data, the data must be transmitted in secure communication channels.

Every mobile agent must first be authenticated to identify whom it represents. The Concordia SecureAgent supports user authentication by using the username/password pairs. Once identified, mobile agents can be checked against the security policy to see whether they are authorized to do certain things at a server. The Concordia Administrator, as shown in Fig. 6.4, provides a user-friendly interface for server, security and service management.

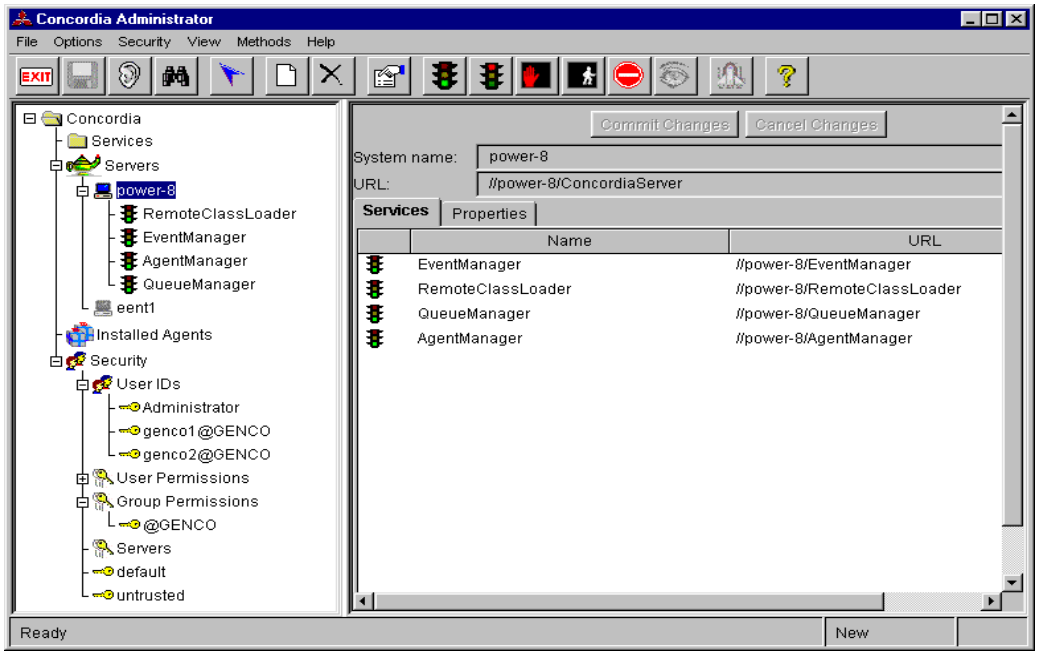

Fig. 6.4 Concordia Administrator

The Concordia server can control the agent's access to resources depending on both the user identification and the server permits. A user can be created for each GENCO. Alternatively, a GENCO group can be used to represent all the GENCOs. In Fig. 6.4, a GENCO group has been created, which has two members: genco1 and genco2. The GENCO group has the permissions to access maintenance-related services.

The user interface to assign different types of permissions to users or groups is shown in Fig. 6.5. The permissions are divided into different groups for agent, class, event, file, etc. For example, the permissions in agent group decide if an agent with the user/group's identity can arrive or be launched to/from this server. The file group permissions determine whether a user or group can access the local files.

Secure communication channels among the mobile agent servers may become important, especially when the data passes through the public network or wireless channels. Concordia mobile agent software provides options to encrypt the data when it is in transit, thus preventing others from tampering with the data. A digital envelope is used to protect the Concordia SealedAgent when traveling.

All the above security measures are supported by Concordia software directly, thus greatly simplify the programming work. There are some other security-related features provided in Concordia. As an example, Concordia mobile agents can work with firewalls, which is important when access to the company Intranet from outside public Internet is needed.

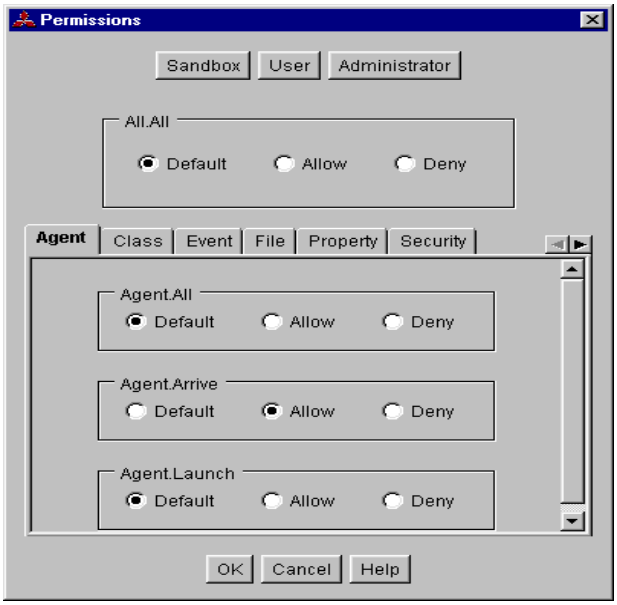

Fig. 6.5 Set user permissions

6.4.2 Interfacing with Other Systems

Mobile agents travel to Concordia servers and call the exposed services to accomplish their tasks. The recommended way to expose services in Concordia is to use service bridges, which are application specific services written in Java and installed at the servers. Service bridges can be thought of as the non-mobile component of a mobile agent-based application. They represent server-specific resources that can be accessed by mobile agents using the exposed interfaces. Service bridges may also act as interfaces for accessing existing systems via RMI, JNI, CORBA, or DCOM.

When the mobile agents need to access the functions of some existing systems not implemented using Java, two possible interfacing techniques are Java Native Interface (JNI) and Common Object Request Broker Architecture (CORBA) [13]. As an example, in the maintenance-scheduling problem, the mobile agent needs to call the Linear Programming (LP) solver and the Mixed Integer Programming (MIP) solver of some optimization packages. Both JNI and CORBA approaches have been implemented to call the optimization library.

In the JNI approach, the optimization routines provided at both the GENCO computer and the ISO computer are stubs to Matlab optimization routines. The linear programming routine of Matlab optimization toolbox is used to solve the ISO sub-problem, and the master problem is solved by simply enumerating all the combinations. As shown in Fig. 6.6, the mobile agent is calling the optimization routines through the Jmatlink [14] module, which uses Java Native Interface (JNI) to invoke the Matlab engine. In this way, a virtual link between the mobile agent and the optimization routines is established.

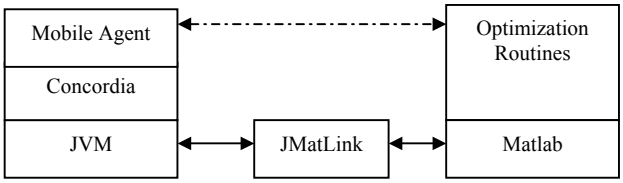

Fig. 6.6 Links between JVM and Matlab

In the CORBA approach, the commercial solver CPLEX 6.5 has been encapsulated using CORBA. CPLEX can solve large-scale LP problems and MIP problems [15]. Using CPLEX, larger systems can be solved since it provides a better MIP solver and can handle larger scale problems. This approach is shown in Fig. 6.7.

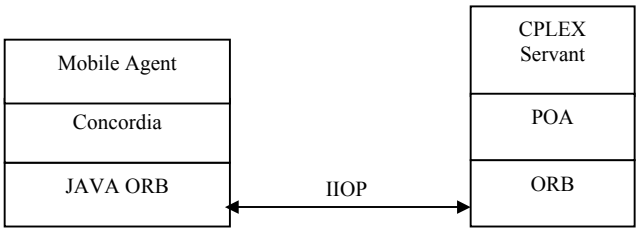

Fig. 6.7 Calling CPLEX using IIOP

CORBA is a platform and language-neutral standard for component programming [13]. CORBA is a standard; the actual software is called ORB (Object Request Broker), which is in charge of locating the remote objects and doing the remote method invocation. The standard protocol IIOP (Internet Inter-ORB protocol) in CORBA specifications ensures the interoperability among the ORBs implemented by different venders. OmniORB [16] is a free ORB and has been used to encapsulate and export the CPLEX library.

Java 2 comes with a Java ORB and supports RMI-over-IIOP directly. Since the current version of Concordia does not work with Java 2 (JDK 1.2.x, 1.3.x) very well, we used another free Java ORB – JacORB [17], which can work with Java 1 (JDK 1.1.x). According to the vendor, the next version of Concordia will work with Java 2.

Exposing the functions of the optimization package (CPLEX in this specific case) using CORBA makes it accessible to other systems, which may be programmed with different types of languages than the one used in the optimization package. In addition, the location of the services becomes transparent. Unlike the JNI approach, where the exposed functions can only be assessed by local Java applications, the CORBA approach makes the functions accessible to more applications with different languages and even from remote computers. For example, in our case, the CPLEX library does not need to be at the same computer as the Concordia server. Since the CPLEX license limits it usage only at the computer where it has been installed, having a CORBA server running on that computer will make it accessible to more applications, thus improving its utilization rate. In this way, fewer licenses need to be purchased.

Using CORBA to call the CPLEX library also provides an example on how to integrate a legacy system using mobile agents. Legacy systems may be implemented using different languages (COBOL, Fortran, ADA, C, etc.) and running on different platforms (e.g., mainframe, minicomputer, etc.). As long as the functions of the legacy systems can be exported using CORBA, those functions can be accessible from Java, and thus by mobile agents.

6.5 Example

A three-bus system is used to demonstrate the main concepts, and the system is shown in Fig. 6.8. A GENCO owns all the three generators and an ISO is in charge of scheduling the transactions over the three lines. Only one maintenance period is considered, and, in that period, at least one generator will be maintained. A transport model is used to check the feasibility of a schedule in the ISO, so the data of line capacity is sufficient, which is shown in Fig. 6.8. The generator data is given in Table 6.1. To further simplify the problem, only the maintenance cost is considered in the objective. Therefore, the maintenance-scheduling problem of (1) is simplified to:

$$Min \quad \sum_{i=1,2,3} \{C_i(1-x_i)\} \tag{13}$$

S.T.

(1) Maintenance constraints: (14)

$x_1 + x_2 + x_3 \le 2$

$x_i = 0$ or 1, $i = 1,2,3$

(2) System constraints (15)

$-f_{12} - f_{13} + g_1 = d_1$

$-f_{23} - f_{12} + g_2 = d_2$

$f_{13} + f_{23} + g_3 = d_3$

$0.5 \le g_1 \le 2.5$

$0.6 \le g_2 \le 2.5$

$0.6 \le g_3 \le 3.0$

$-0.5 \le f_{12} \le 0.5$

$-0.5 \le f_{13} \le 0.5$

$-1.0 \le f_{23} \le 1.0$

Table 6.1 Generator Data

Unit	Min cap. (p.u.)	Max cap. (p.u.)	Maint. cost *Ci* ($)
1	0.5	2.5	300
2	0.6	2.5	200
3	0.6	3.0	100

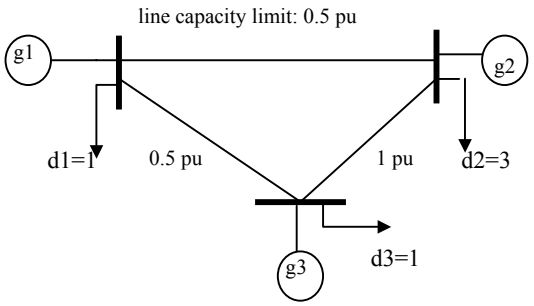

Fig. 6.8 Three-bus system

The above problem can be decomposed into a master problem solved at the GENCO, and a sub-problem checked at the ISO. The master problem tries to minimize the maintenance cost and consists of the objective and the maintenance constraints part, while the sub-problem is only responsible for the system constraints (only considering the overload of the transmission lines in this example).

The computer setup has been shown in Fig. 6.3 and the mobile agent will travel between those two computers. The mobile agent calls some service functions provided by the GENCO computer at first to solve the master problem. After a trial schedule is obtained, the agent travels to the ISO computer with the schedule and submits it. The services provided at the ISO computer check the schedule and generate an infeasible cut if it is infeasible. The agent brings the cut back to the GENCO computer and adds it into the set of constraints, and then starts the next iteration. The process continues until no cut condition is returned from the ISO computer.

At the initial step, the mobile agent's itinerary contains three destinations: the GENCO computer, the ISO computer, and then back to the GENCO computer. According to the status of problem, the mobile agent will modify its itinerary dynamically. When the ISO sub-problem returns a cut, the mobile agent resets its itinerary and begins the next iteration. The results returned by the mobile agent at each iteration are given in Table 6.2.

Table 6.2 Results of Each Iteration

Iteration	Problem	Results
1	Master	x = [1, 1, 0], z=100
	Sub	Benders cut: c*x ≤ -2.9, c = [0, -2.499, -2.999]
2	Master	x = [1, 0, 1], z= 200
	Sub	Benders cut: c*x ≤ -1.4, c = [0, -2.499, 0]
3	Master	x = [0, 1, 1], z= 300
	Sub	No Benders cut, so the above schedule is feasible and approved

From the above table, we can see the total cost is 100 and unit #3 will be taken out for maintenance in the first iteration. Since this case does not consider the system constraints, the solution will not be feasible when it is submitted to the ISO computer. An infeasible cut is returned, which is $-2.499*x2 - 2.499*x3 \leq -2.9$ as shown in the table. By adding this constraint into the master problem, the scheduling problem is solved again at the GENCO computer, and the result is that unit #2 will be maintained this time. Again, this result is submitted to the ISO computer and with a new cut returned. After adding this new cut, the solution turns out that unit #1 will be maintained and the total cost is 300. This time, ISO computer does not return any cut, which means this schedule is feasible and thus approved.

Both approaches proposed in Section 6.4 have been implemented and compared. The results are the same and no difference in speed has been observed. The main reason is that the system is very small. When bigger systems are used, it can be expected that using CPLEX library instead of calling Matlab will give much better performance.

This example illustrates one possible application of mobile agent software in the maintenance-scheduling problem. Actually, the GENCO may use a very complicated method or system to schedule the maintenance. In addition, the ISO may consider many more factors when checking the feasibility of a submitted schedule. However, the iteration process among different entities (the GENCO and the ISO in this case) is unchanged.

7. Distributed Information Processing

7.1 Heterogeneous Data Access Problems and Requirements

During maintenance or repair work, the maintenance crew will need to access information distributed across the utility and stored using different data formats. Several possible locations of the information storage are the enterprise maintenance system, the substation data concentrators and the maintenance crew's computer.

The information about spare parts, test procedures, historical maintenance records, and instruction manuals, etc. is typically accessible in the enterprise maintenance system. In addition, the enterprise maintenance will usually utilize a RCM or conventional maintenance scheduling system to generate work orders. The work orders indicate when and where to perform what kind of maintenance on which devices.

The information about substation equipment may be retrieved from the substation computers or concentrators. With the introduction of continuous monitoring of circuit breakers, real-time data becomes available for accessing in the substation concentrators. The continuous monitoring instrument may measure the coil current profiles and switching timing during the normal operation. The condition of a circuit breaker can be assessed using some signal processing and artificial intelligence techniques. In this way, the time-directed preventive maintenance may be replaced by condition-directed predictive maintenance. The real-time data in the substation concentrators is also a useful complement to the historical information stored in the enterprise maintenance system. The data may be utilized to automatically update or populate the enterprise maintenance database.

The maintenance crew may have the inspection or test report stored on a mobile computer. The crew may need to update the status of the work order stored on the computer as well.

Since the maintenance information is distributed among different heterogeneous systems, a software technique that has the flexibility of interfacing with multiple heterogeneous information systems is desired. The software should have the following characteristics to meet the maintenance information exchange requirements:

- Security support on public networks (encrypted data transmission, user authentication and authorization)
- Efficient network bandwidth usage
- Robust and fault-tolerant communication in an unreliable environment using portable personal communication devices
- Ability to integrate with heterogeneous systems
- Automatic software update to ease the user burden.

Equipping the crew with new information access methods to replace the old paper-based information exchange and logging method may improve efficiency because less time will be spent on preparation, reporting and logging.

7.2 Mobile Agents for Distributed Information Processing

The unique features of mobile agents make them suitable for the tasks of monitoring and maintaining power system apparatus. The information involving data recording and maintenance is distributed among the utility facilities and even different companies. The mobile agent can travel to the place where the data is stored to do the processing task locally. Thus, the use of computing resources and network resources can be minimized. Compared with distributed database systems, the mobile agent method can integrate heterogeneous database systems and is more suitable when the needed information requires searching multiple databases of different entities. Concordia collaborating agents provide the framework for combining the search results.

Fig. 7.1 illustrates an application scenario for a circuit breaker monitoring application. The monitoring devices record the circuit breaker status snapshots and transfer them to the substation concentrator. An AI application can run on the concentrator and detect any abnormality. The application can utilize the historical information and the status snapshot to create a maintenance schedule or repair request. The concentrator may launch agents when necessary to do the data search.

Using notebooks, handheld computers or even cell phones, the maintenance crew can launch an agent to do a selected statistical analysis to aid the device diagnosis. They may also search the enterprise inventory database when parts need to be replaced. A purchase request can be placed when the parts are out of stock. The crew should be notified when the parts become available. Furthermore, the maintenance crew can be notified when emergency events happen. Handheld computer (e.g., Palm and Visor) will be supported through communication nodes or may have Java support directly.

During the work process, the window indicated in Fig. 7.1 will show all the activities of the mobile agents with animation. It will help the user understand what the agents are doing.

In the following sections, we will introduce a few typical application scenarios where the mobile agent software may be helpful.

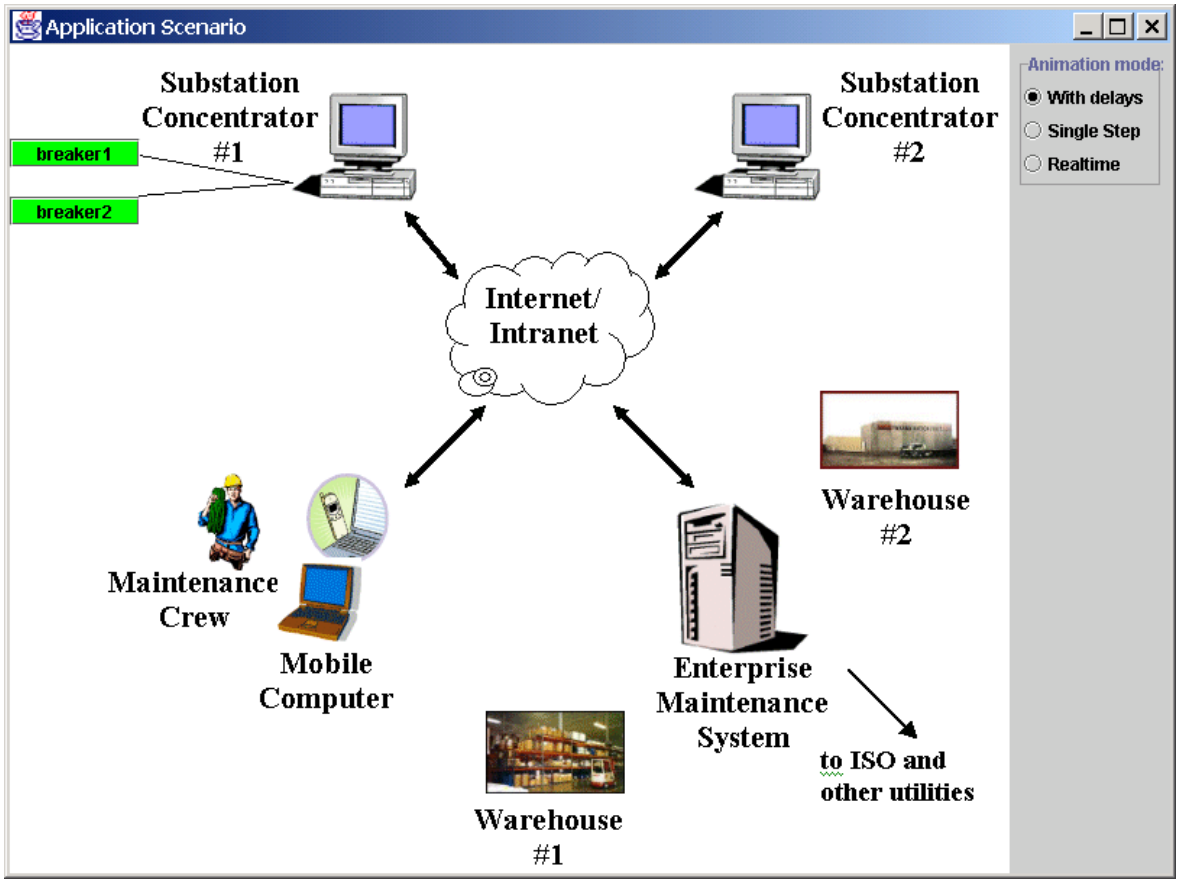

Fig. 7.1 Mobile Agent in monitoring and maintenance of power system devices

7.2.1 Retrieving and Updating Work Orders

First, the maintenance crew needs to get the work orders from the enterprise maintenance system. The work orders may be generated using the RCM system or based on some heuristic rules. As shown in Fig. 7.2, by clicking on the "**Work Orders**" node, a *Workorder* agent will be sent out to the maintenance system to retrieve the work orders. The mobile agents have the flexibility to interface with different types of maintenance scheduling systems. If a different scheduling system is used, only the *Workorder* agent needs to be updated.

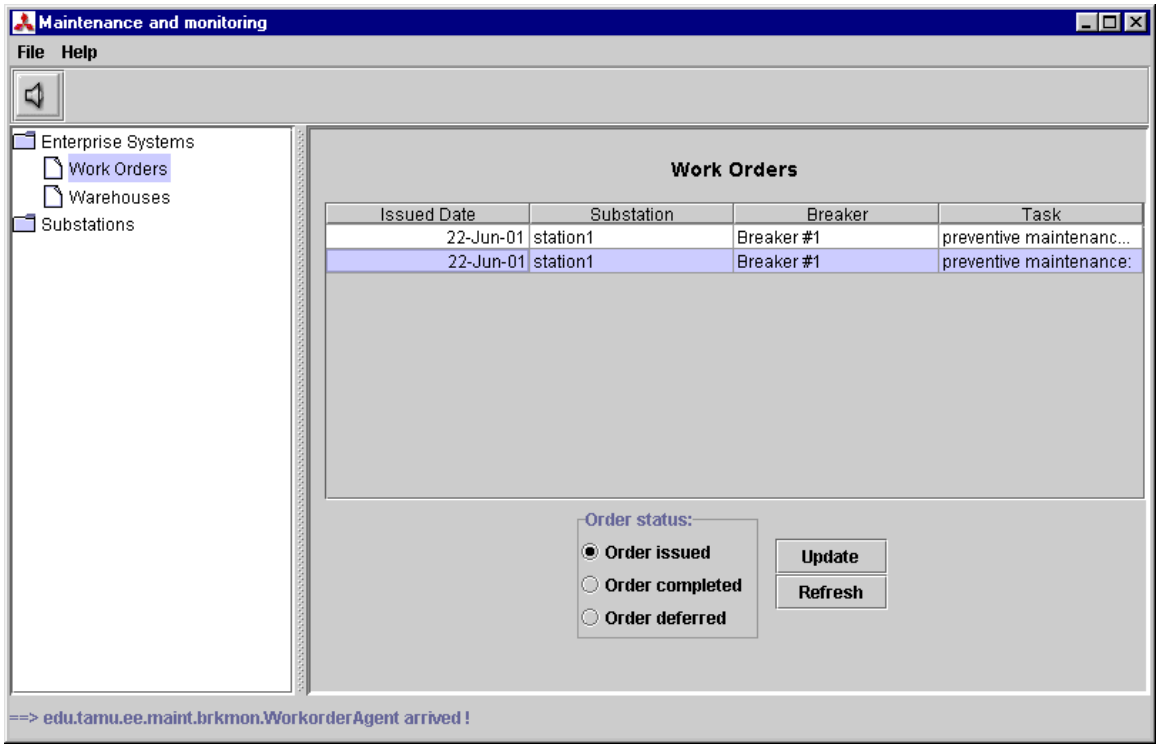

Fig. 7.2 Retrieving work orders

As shown in Fig. 7.2, after the work orders have been retrieved, the maintenance crew will know where and when to perform the work. When the maintenance work is finished, the crew can update the status of the work order.

7.2.2 Retrieving Substation and Circuit Breaker Information

During the maintenance process, the maintenance crew may need to access information about the substations and circuit breakers. As shown in Fig. 7.3, by clicking on the "Substations" node, a Substation agent will be sent to the maintenance system to get the necessary information about the substations. In addition, as shown in Fig. 7.3, information is available about a substation, such as type, voltage level, and its single line diagram.

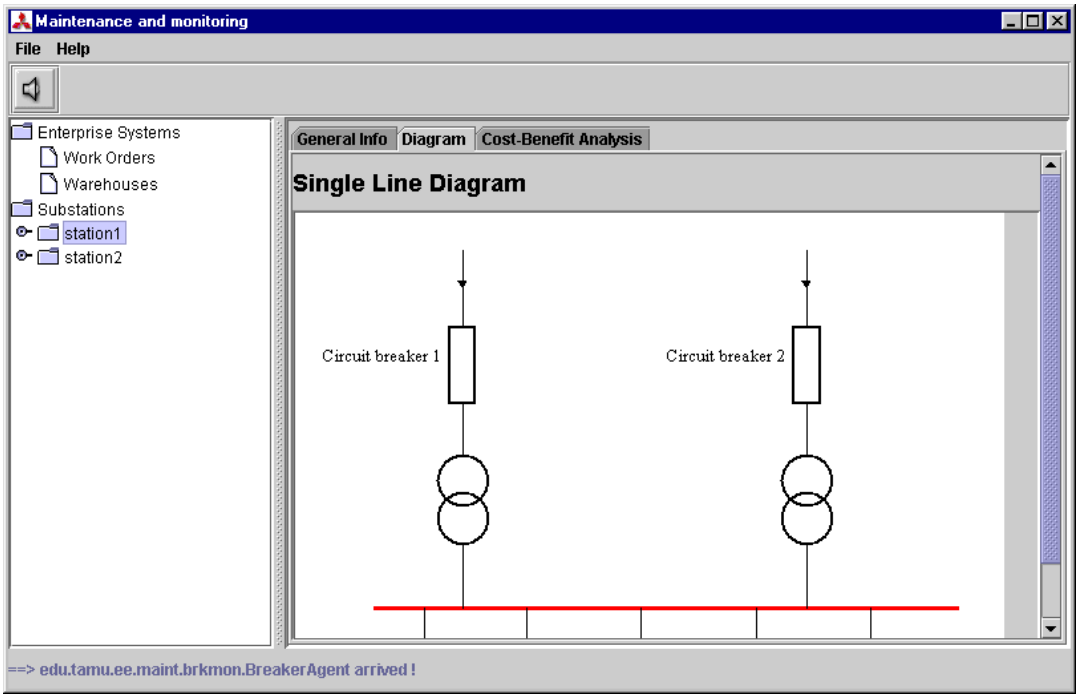

Fig. 7.3 Retrieving substation information

As shown in Fig. 7.4, by clicking on the substation of interest, a mobile agent will be sent out to collect information about circuit breakers in that substation.

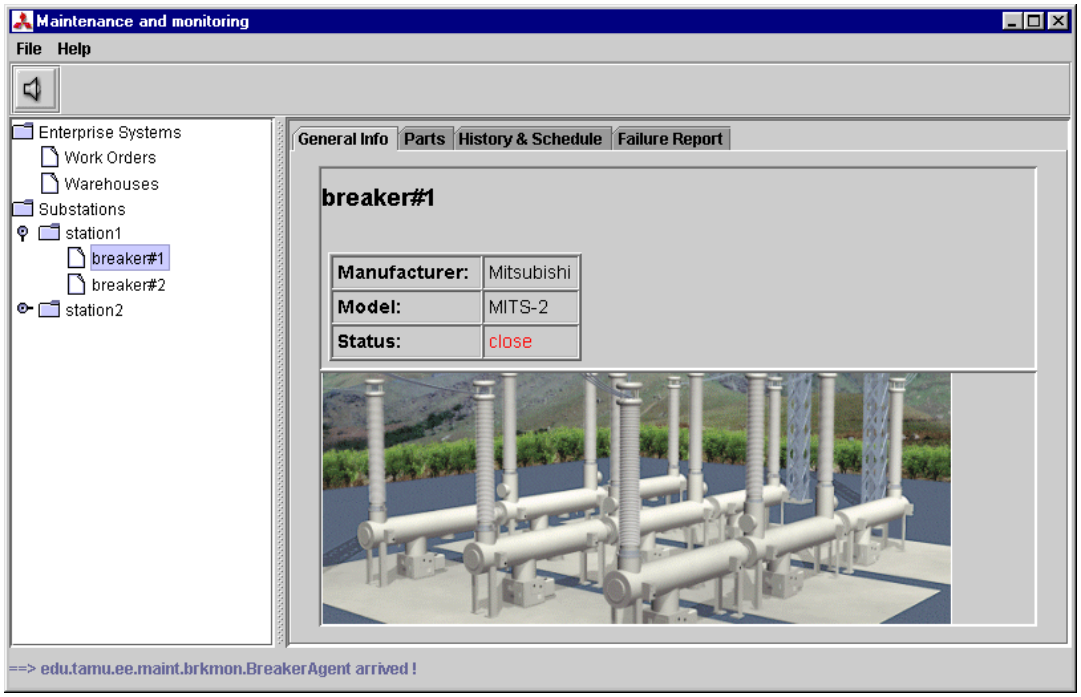

Fig. 7.4 Retrieving circuit breaker information

First, the mobile agent goes to the substation concentrator to collect the circuit breaker model and status, then it will go to the maintenance system to retrieve the information related to this specific breaker. In addition, the maintenance history for this breaker will be retrieved from the maintenance system.

7.2.3 Monitoring Circuit Breaker Events

The crew can select the events of interest inside a substation to be monitored. When those events happen, the crew will be notified.

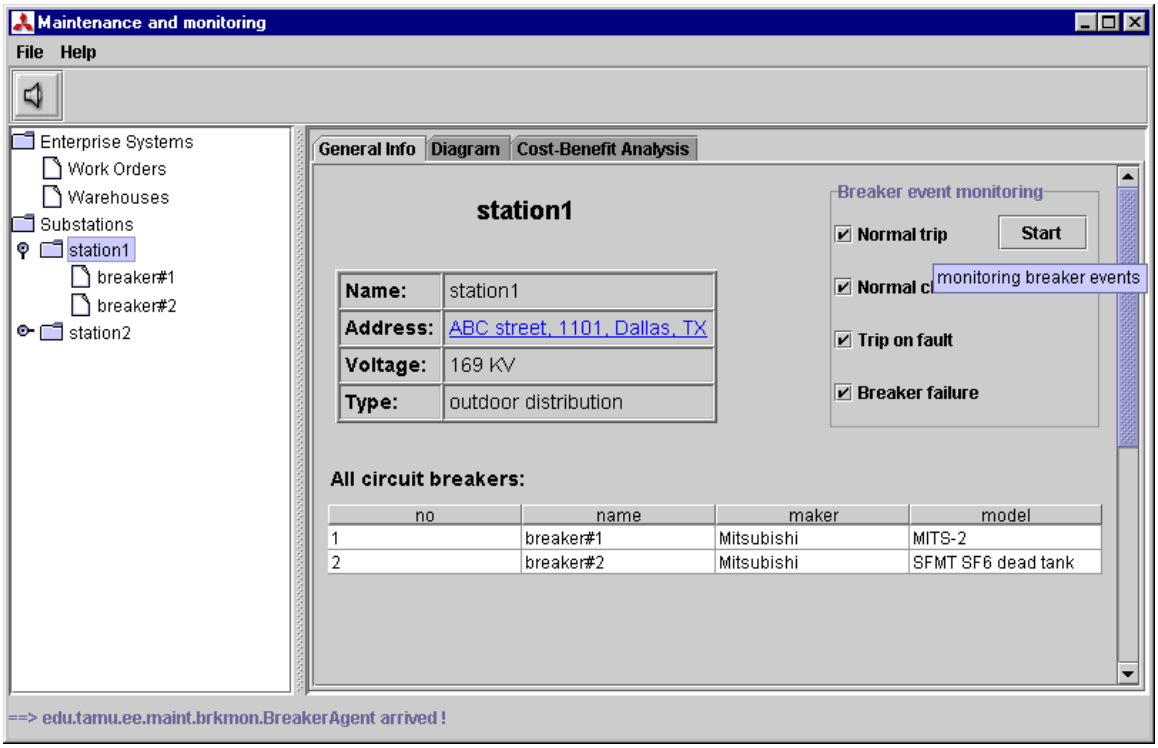

Fig. 7.5 Event monitoring

Concordia's distributed event feature is useful for the device monitoring and event notification application. The maintenance crew can select the events of interest to monitor.

7.2.4 Retrieving Information about Spare Parts

During the preventive maintenance process, when some parts are found to be worn and in need of replacement, the crew will need detailed information about spare parts. By clicking on the "Warehouse" node, a mobile agent will retrieve information from the warehouse. The crew can select a warehouse, and its map will be shown.

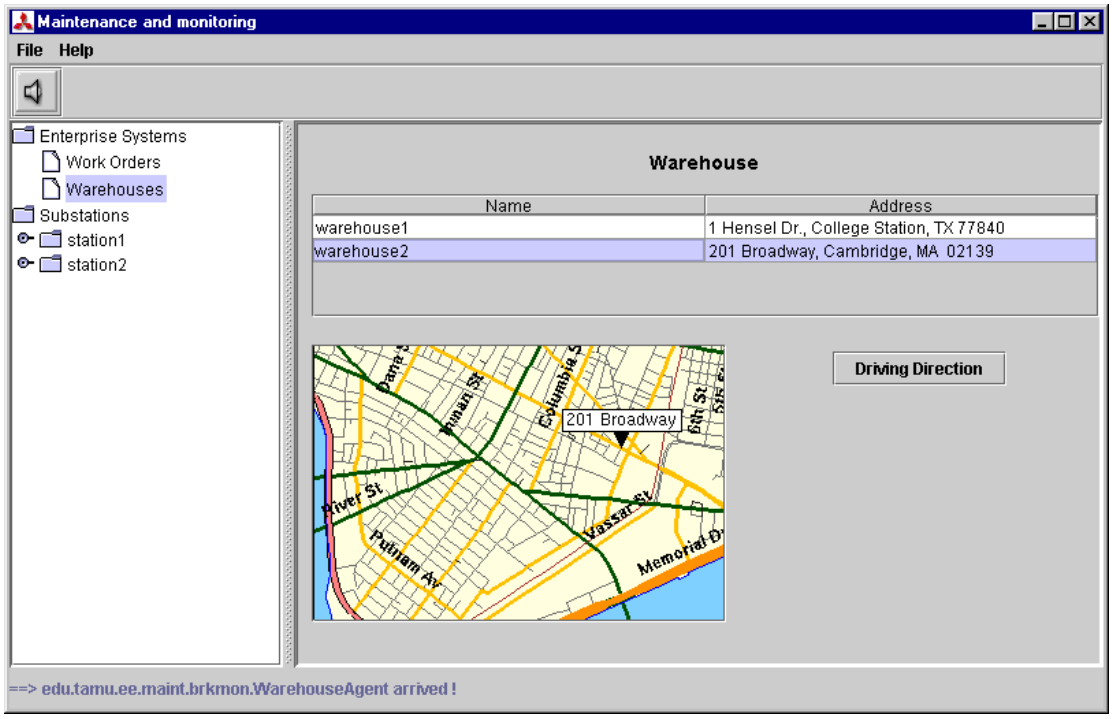

Fig. 7.6 Warehouse information

By clicking on the "parts" tab, the maintenance worker can see all the parts of the breaker. When a part is selected, a "parts" agent searches each warehouse until the required part is located.

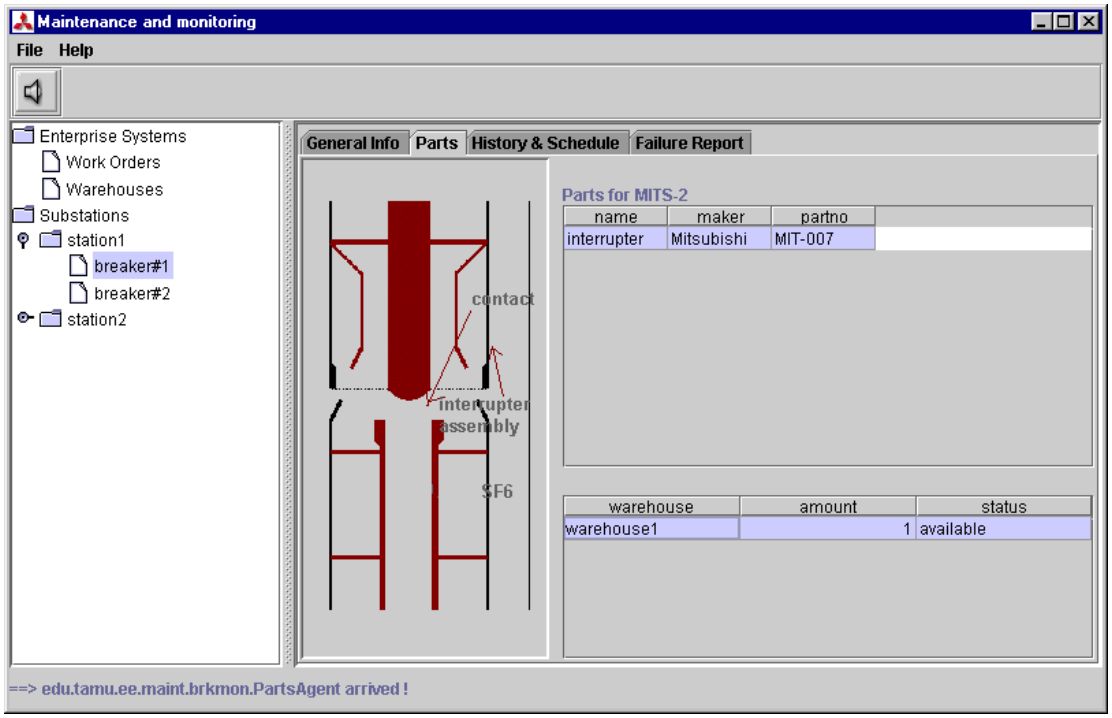

Fig. 7.7 Parts information of a specific circuit breaker

7.2.5 Preparing Reports

Agents may also be used to prepare reports. When a report is requested, mobile agents are dispatched to the substation and the maintenance system to retrieve data relevant to the report. The agents then collaborate to produce a report in XML format.

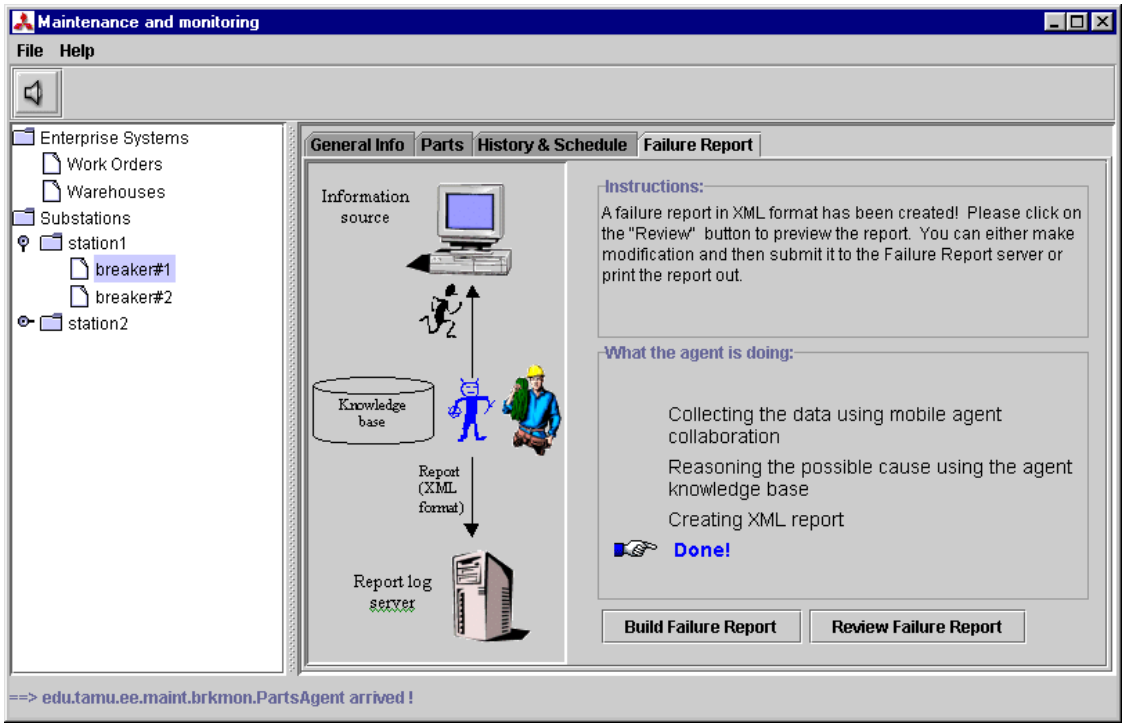

Fig. 7.8 Preparing IEEE compliant failure report

The created report is shown in Fig. 7.9. After reviewing the initial report, the maintenance crew can print it out or submit it to the report server directly.

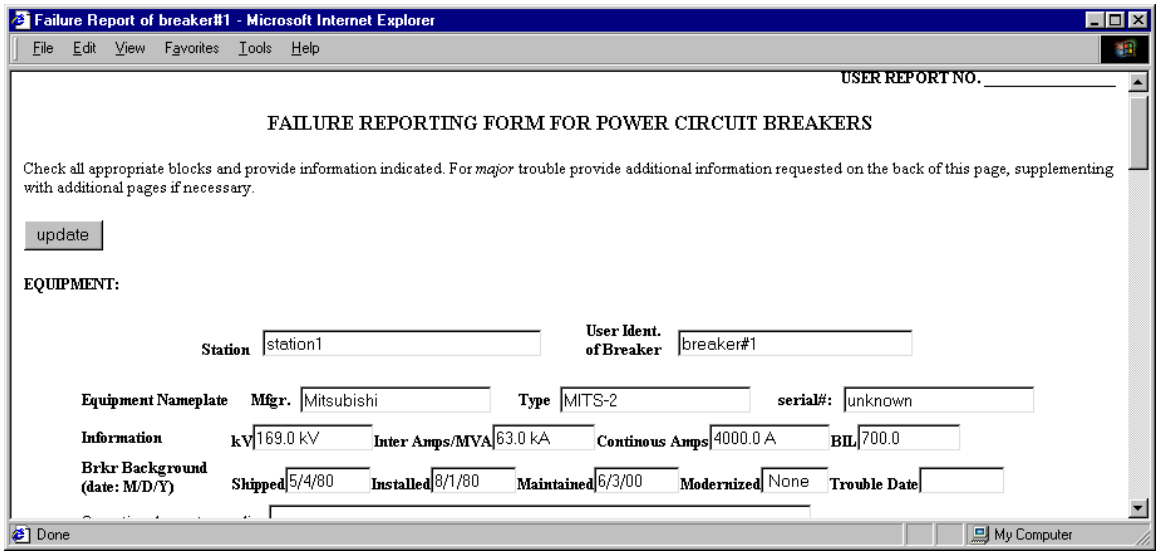

Fig. 7.9 Failure report created by mobile agents

8. Integrating with Circuit Breaker Monitoring Software

8.1 Circuit Breaker Monitoring Software

The Automated Circuit Breaker Monitoring project is a non-PSerc project funded by CenterPoint Energy and carried out by Texas A&M University. Prototype software has already been created [18]. It consists of three modules: (1) data acquisition module, which collects data and transfers it to the data concentrator computer; (2) signal analysis module, which is in charge of extracting parameters and/or features from the recorded data; and (3) expert system module, which infers the condition of the circuit breakers using the extracted features. Here we are only interested in the last two modules. The system structure is depicted in Fig. 8.1. The dotted lines represent the possible interfaces.

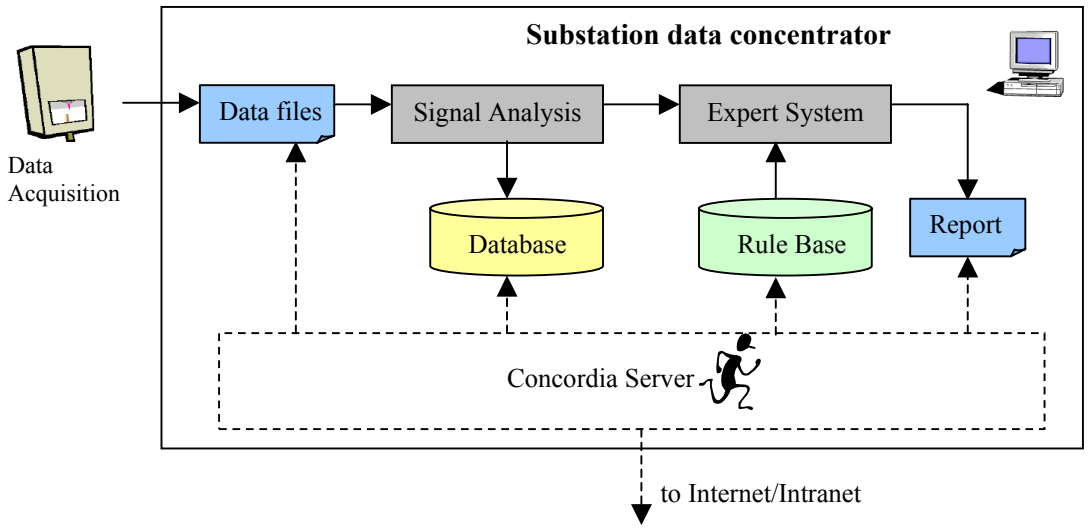

Fig. 8.1 Structure of circuit breaker monitoring software

The signal analysis module is a standalone program made in C++, and the expert system module is made using CLIPS [10]. The signal analysis module writes the results into a database, but since the results cannot be easily accessible from CLIPS, temporary files are used to exchange information. When the expert system module is invoked, it will create a brief report about the condition of the circuit breaker. This report, the extracted parameters, and the original recorded data files may be of interest to the maintenance staff. Those data may be applied for scheduling work orders, analyzing device failures, etc.

In summary, the online monitoring system and software make the following real-time data available: recorded data files, extracted parameters, and analysis reports. We will apply mobile agent software to better utilize and distribute that information.

8.2 Integration

We have built a prototype system to demonstrate the event registration and notification scenario. As shown in Fig 8.2, the maintenance staff and systems can subscribe to the events in which they have an interest. The Automated Circuit Breaker Monitoring software will post a notification when an event happens. The server will check against its subscriber list and distribute the notification to those who are interested.

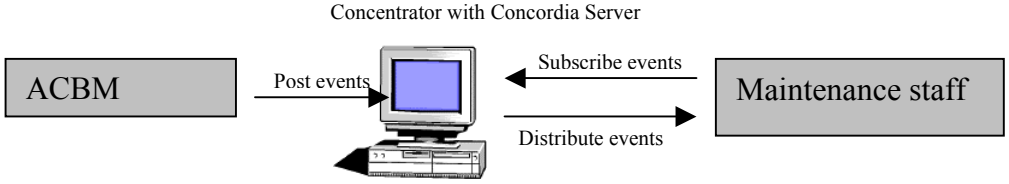

Fig. 8.2 Event subscribing and publishing

In Fig 8.3, the user can select the type of events to be monitored. When the events of interest happen, the user will be notified. Those events may be used to initiate a maintenance task or are simply logged in a historical database.

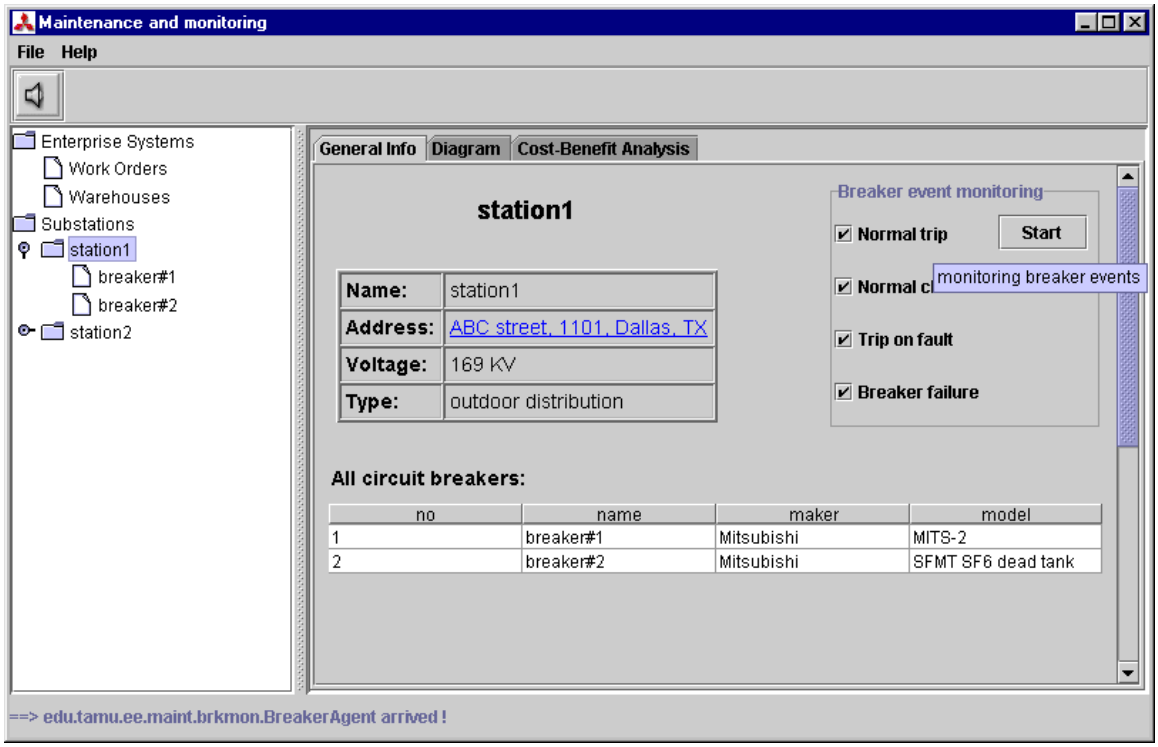

Fig. 8.3 User interface for selecting events of interest

Sometimes, for analyzing or logging purposes, the user or system may need to access more information about the event. One example is to retrieve COMTRADE files related to the event. Here, we utilize mobile agents to fetch the files.

When the user double clicks on the event of interest, a user interface of Fig. 8.4 will appear. The user has the option to retrieve various data. The "Event report", which is generated by the expert system, contains only summary or diagnostic information about the event. The "Event parameters" data give the information describing various aspects of the event. Finally, the "Event raw waveforms" data created by the data acquisition module can be retrieved so that detailed analysis can be conducted.

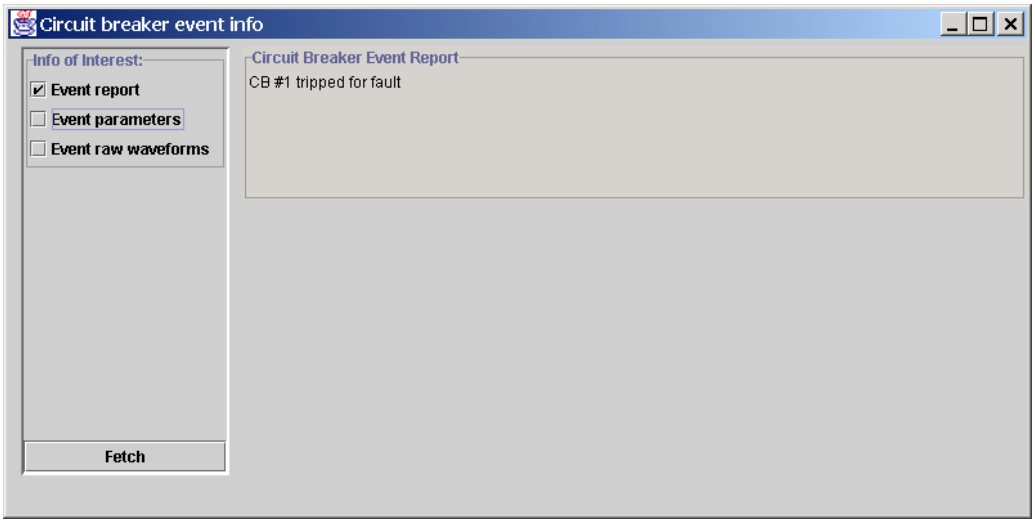

Fig. 8.4 User interface for retrieving the COMTRADE files

Retrieving the whole waveform file may consume considerable network bandwidth when the file is large. To minimize the network bandwidth usage, the mobile agent will compress the files using the ZIP algorithm supported by the Java platform. Further software development may introduce wavelet compression. Its high compression ratio may be desirable, although the wavelet compression is not lossless.

Once the waveform file has been fetched, the user can view the file, as shown in Fig. 8.5. Further software development may include zooming and selected displaying, etc.

Jess, the Java version CLIPS engine, has been utilized to run the expert system on the substation concentrator. Unlike CLIPS, whose support has been discontinued by NASA, the development of Jess is continuing. New versions keep coming out. An advantage of Jess is that it is implemented in Java, which makes its integration with the Concordia software very easy. With the JDBC support to access all kinds of databases and the Jess engine to utilize the rule base, the interfacing problem between the signal analysis module and the expert system module can be eliminated. The mobile agent can retrieve parameters calculated by the signal analysis module from the database, and build the corresponding facts. The inference can be implemented by calling the Jess engine, and passing the facts and existing rule base. The results are saved in memory, and can be printed out as a report or brought back directly to the user.

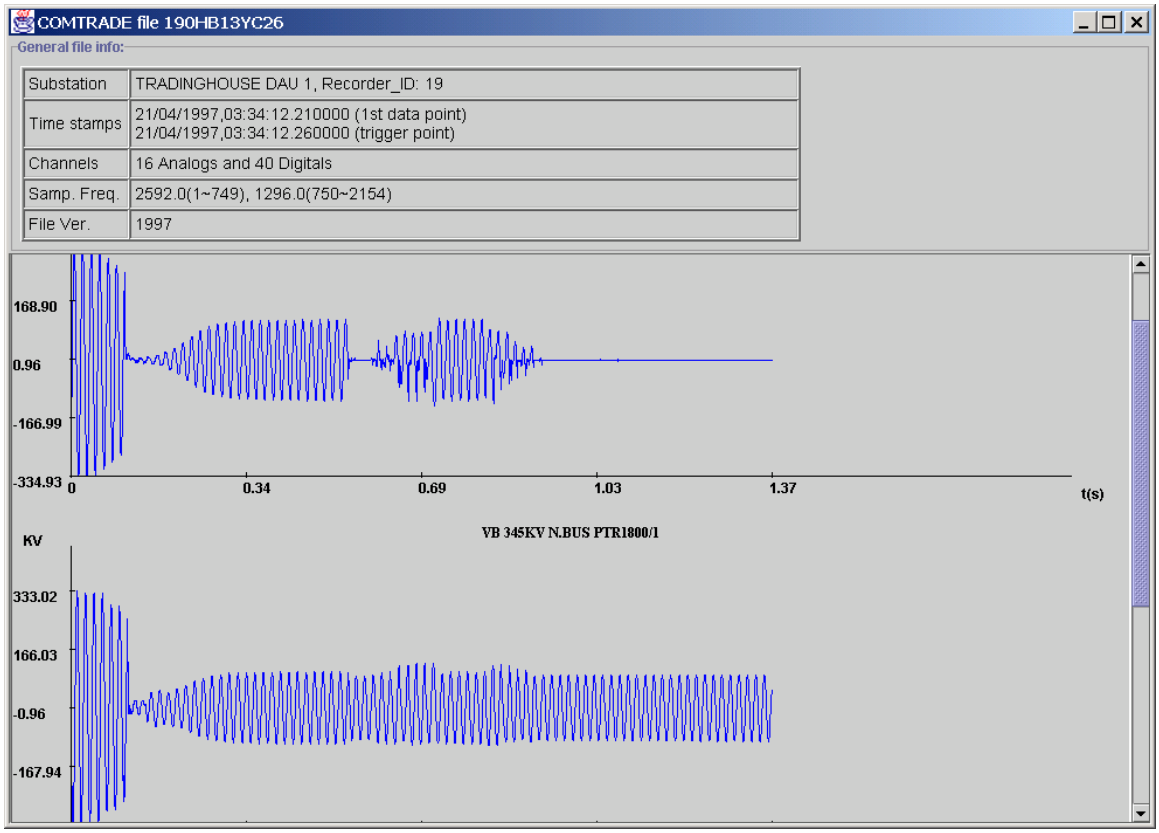

Fig. 8.5 Viewing the COMTRADE file

As shown in Fig. 8.6, the mobile agent may need to fetch the report, the extracted event parameters, and/or the raw waveforms. When the report is required, the mobile agent will read in the facts from the database and the rules from the rule base, and then invoke the Jess inference engine. The Jess engine will create the report. For event parameters, the mobile agent can query directly against the database using JDBC. Furthermore, the mobile agent can access the local file system of the substation concentrator to retrieve data files.

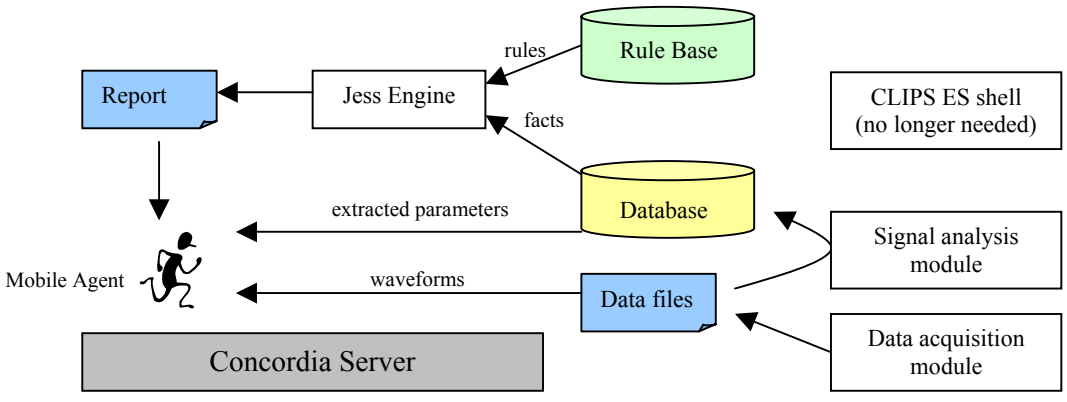

Fig. 8.6 Data flow inside the substation concentrator

9. Agent-Oriented Workflow Management

The application of mobile agent techniques described in chapter 7 was mainly focusing on retrieving data and integrating system tasks. In this chapter, we will explore a new approach that will fully utilize the functionality of software agents.

An agent (as shown in Fig. 9.1) should be able to perceive environment changes via sensors and take appropriate actions via its effectors [19]. Since only software agents are discussed here, we use events to model percepts, and agent methods for actions. An agent can subscribe to events of interest, which is equivalent to "percepting the environment". The event handler will translate those events to facts and insert them into the inference engine. The engine will determine what the agent should do. The agent provides callback functions to the inference engine. Those callback functions represent the actions that the agents support. Actions may affect the environment, which is modeled by triggering events.

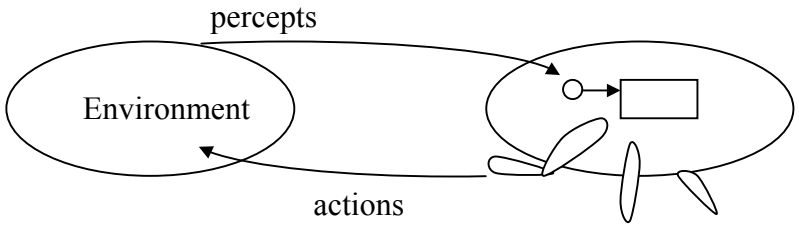

Fig. 9.1 Agent interact with environments though sensors and effectors [19]

9.1 Function Requirements for Workflow Management

Workflow management deals with the execution of business processes. Typical workflows are executed in distributed computing environments such as in large enterprises. Therefore, workflow management systems must be able to deal with distribution and heterogeneity [20]. Distributed systems such as CORBA, are often used to implement workflow management systems [20].

In addition, certain business processes may become very complicated. To manage the workflows of those business processes, careful planning and coordinating are often required. A method that can utilize expertise to help user managing the workflow is desired.

9.2 Agent-Oriented Approach

Like the Object-Oriented approach where objects are the basic modeling unit, the Agent-Oriented approach uses agents as the basic entities to model the systems. However, the Agent-Oriented approach is not as widely used as Object-Oriented modeling (including Object-Oriented Analysis (OOA), Object-Oriented Design (OOD), and Object-Oriented Programming (OOP)

[21]) A distinct difference between objects and software agents is that software agents are autonomous active components whereas objects are usually passive. Objects encapsulate their internal details and provide services for manipulation by outside users . External processes usually control the life span of objects by creating or deleting the objects. Conversely, software agents are autonomous components; they control their own life span. Furthermore, software agents usually act automatically when environment changes are detected rather than being commanded by an external process.

As we can see, objects are good for modeling passive processing applications that mainly provide services to the outside world. For active processing applications, software agents may be a more natural metaphor due to their autonomous characteristic. The mobility aspect of software agents makes them advantageous in distributed heterogeneous environments, and the intelligence aspect adds more power to the software agents, making them act rationally. Agents may also provide better user interfaces.

As an example, we will look at the work order management scenario in device maintenance. In this scenario, the maintenance schedule system will issue a work order and the maintenance crew will try to accomplish the order and report the results. If we look at the scenario in a data-oriented way, we will try to retrieve the work order to the maintenance crew computer. We can use an object to represent a work order; it will have internal data members representing the work order and functions to update those internal data. The maintenance crew will look at the order and make plans to fulfill the order. They may need to retrieve different data, perform a test, get spare parts or replace worn ones. Finally, the crew will file a report for all the activities. Thus, an external control process (driven by the crew), which models the workflow to finish the work order, is still needed.

How would this work order management scenario change is mobile agents were used? The agent will facilitate the whole process of searching the information needed for issuing a work order, verifying availability of the spare parts, scheduling the work, and automating the reporting and logging processes after the work is done. By changing our viewpoint to an agent-oriented way, we will have only a work order agent that will plan and direct the workflow until it is finished. Instead of retrieving the work order to the maintenance crew computer, a work order agent will travel to the crew computer. The work order agent is generated from the maintenance scheduling system. It has the knowledge related to the completeness of the work order. Once the agent arrives, it will inform the crew what should be done. The agent will also update the workflow according to the current situation provided by the crew. Finally, when the work is finished, the work order agent will travel back to the maintenance system to submit the report automatically. As compared to the old data-driven method, the new approach is friendlier to users. In addition, a software agent, with the control process built in along with the representation data for the work order, is a more natural model in this specific case.

Usually every management subsystem may provide several primary activities that can be organized in different ways to accomplish basic workflow. A workflow may need modification according to the results of the activities. We can have different types of agents to represent different workflow. By sending out different mobile agents, different workflow can be achieved using the same set of distributed information sources.

In the next section, we will use a software agent to model a work order. The workflow is modeled using production rules.

9.3 Agent-Based Workflow Management

Usually, the work order is issued by a maintenance management system. The work order will depend on the types of maintenance practices that have been adopted. Once an order is issued, the feasibility of the work order will be checked. For example, if a certain maintenance task requires replacing parts, then those parts must be available. Otherwise, the work order must be classified as pending, waiting parts. Then, the work order will be dispatched to an available maintenance crew. After receiving the order, the crew may need to pick up testing equipment or parts before going to the substation to perform the actual maintenance work. After the maintenance work is done, reports may be required.

An example of the maintenance workflow is given in Fig. 9.2.

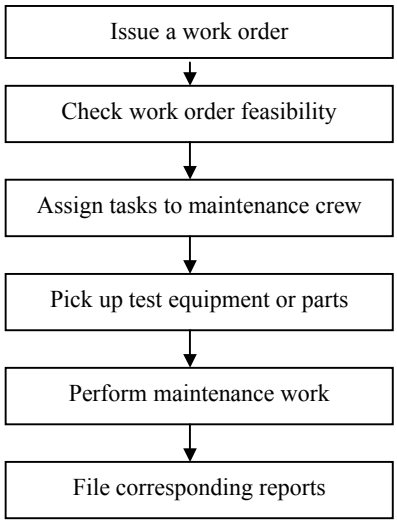

Fig. 9.2 An example of maintenance workflow

We use IF-THEN rules to model the workflow of Fig. 9.2. The antecedent part of the rule gives the requirements to accomplish one step of the workflow, and the consequent part defines the actual action of the step. An example rule is given in Fig. 9.3.

```
(defrule rule_task_assignment
  (maint_task (taskname ?taskname) (substation ?sta) (breaker ?brk) (taskdate
?taskdate) (tasktype ?tasktype))
  (feasible maint_task ?taskname)
  =>
  (printout t "assigning " ?taskname " to available crew " crlf)
  (agent-callback assignTask ?taskname)
```

Fig. 9.3 An example rule

When all the conditions of a rule are satisfied, the rule will be fired and new actions will be taken, which may bring the workflow to the next stage. If one or several conditions are not met, no rule will fire and the agent will wait for outside events. By subscribing to the events of interest (e.g., new-parts-arrived or work-order-finished), the agent will be able to drive the workflow asynchronously. How the rules control the behavior of the agent is shown in Fig. 9.4. The software agent provides *callback functions* to the rule base for controlling its actions. *Events* are in fact a type of environment perceptions of the software agent.

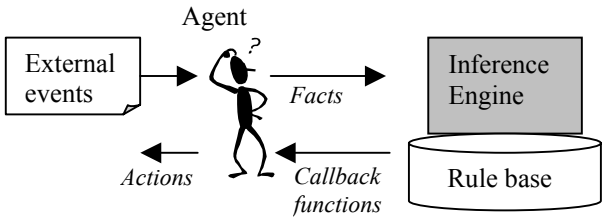

Fig. 9.4 Interaction between rules and agent

A. Issuing a work order

The enterprise maintenance system will create a work order agent for dispatching a work order. The work order agent will first try to locate an available maintenance crew from the crew list and communicate with other work order agents. If all maintenance crews are busy with other work orders, the current work order agent will subscribe to the Work-Order-Finished events at the event server, and then enter the "sleep" status. When a maintenance crew finishes its task, the corresponding work order agent will return to the maintenance system and notify all the subscribers of the Work-Order-Finished event. Then, the work order agent that was in "sleep" status can begin to enter the "ready" status.

In an ideal case, the maintenance crew should have a personal agent that is in charge of arranging its schedules. The work order agent should check against the crew's personal agent for its availability. In the demonstration system, we simply assume a maintenance crew is always available for work.

After selecting a maintenance crew, the work order agent will notify it about the work order. In addition, the agent will record the names of the crew members. When other agents query about the availability of the crew, the agent will be able to give a correct answer.

B. Driving the maintenance workflow

When the work order agent is created, the name of the workflow rule base is provided to the agent. With the knowledge about the maintenance workflow, the agent knows the procedures for finishing the given work order. Therefore, the agent will be in charge of driving the maintenance workflow.

First, according to the type of maintenance task, the agent may need to check the availability of the parts of a certain circuit breaker model. The agent will check each warehouse until the required parts are located. The checking sequence of the warehouses is decided according to the distance to the circuit breaker and the maintenance crew. If the required parts are not available, the work order agent will enter "sleep" status and subscribe to the Parts-Arrived events at every warehouse event server. The work order agent will release its maintenance crew due to an inability to advance the workflow. When the parts arrive, the agent will receive notification and will try again to find an available maintenance crew.

After the agent travels to the maintenance crew computer, it will also provide driving directions to the warehouse and substation, along with maps to the crew. The crew will pick up the parts and then drive to the substation.

After the crew arrives at the substation, the agent can show the maintenance crew the one-line diagram of the substation. The agent may retrieve the diagram from the enterprise system or from the substation concentrator. The agent will also display the information about the circuit breaker to be maintained. The information includes model, maintenance history, etc.

More importantly, the agent will show the recommended procedures for maintenance. The agent will try to retrieve the procedure description from the enterprise maintenance system. If some video (or audio) recordings about the maintenance are available, the agent can play the video to the crew. In addition, the agent can check the experience level of the crew. If the crew is very experienced, the display of the maintenance procedure may be skipped.

If the recording device is available, the agent can begin to record the maintenance process when the crew commands it. The video can be used later to instruct new workers.

C. Completing the work order

When the order is finished, the work order agent can help the maintenance crew prepare reports. Usually, the formats of the reports are standardized in a company and the agent can automatically fill in most of the information.

The work order agent will return to the enterprise maintenance system and submit the report. In addition, it will notify the event server with a Work-Order-Finished event. Then, the work order agent will destroy itself and its life-span will end.

9.4 Further Discussion

A. Agent perception and action

Agents need to sense the environment via "sensors" and act through "effectors". We use events as the only means of perception, and the actions are implemented as callback functions. An agent can subscribe to events of interest. The event handler will translate those events to facts and insert them into the inference engine. The engine will determine what the agent should do. The agent exposes callback functions to the inference engine. Those callback functions represent the actions the agents support.

Integration with Microsoft Agent was also explored [22]. Microsoft Agent has speech recognition and synthesis engine for multiple languages that may be usable for interfacing with maintenance crew.

B. Rule base design

A rule base has been used to model the workflow. The software agent provides callback functions to the rules. One criterion on how to design the rule base is that when people want to change an agent's behavior, they can achieve this simply by updating the rules. Otherwise, what is the purpose of having a separate rule base? After all, we can program the rules into the agent. The only drawback is that we need recompile the program after the changes have been made; this is not a big problem when mobile agents are used. Therefore, only a careful function partition between the agent and the rule base can justify the existence of a rule base.

The workflow model using production rules is like an event-driven structure. When a new event happens, the agent accepts the notification and inserts new facts into the inference engine; this may trigger certain rules. A comparison with a goal-driven model may be worth investigating.

C. Mobility

Mobility is desirable since software agents can travel to maintenance crew computers that may not always be connected to the network. Code migration can be divided into weak mobility and strong mobility [12]. In the weak mobility model, only the code (along with initialization data) is transferred. As a result, the transferred program always starts from its initial state. In the systems that support strong mobility, the execution status of the program will also be transferred, so the program can continue its execution when arriving at the new location. The Jess inference engine (*jess.Rete*) implements the *java.io.Serializable* interface which makes it possible to transfer the execution status of the inference engine along with the mobile agent.

Several problems had been noticed during the integration of Concordia with Jess. First, although Jess version 5 implements the *java.io.Serializable* interface, it actually does not serialize all its status. Therefore, it is impossible to use Jess version 5 with mobile agents if it is required to keep agent status. Second, version 6 of Jess uses JDK1.2, while the Concordia software needs JDK1.1 to work properly. Third, the Concordia software has certain problems in transferring a mobile agent when it carries a Jess inference engine (*jess.Rete*). The reason might be due to the complexity of the engine.

A script has been written to convert Jess 6 codes compatible with JDK 1.1. This allows the converted codes to be compiled.

To solve the problem that Concordia cannot serialize *jess.Rete*, the instances of the Jess engine are declared as being transient inside mobile agents and, thus, are not serialized by Concordia. Instead, the engine is manually serialized into a byte array that Concordia does not have problem in transferring. Once the mobile agent arrives at a new host, the engine status is restored from the byte array. The *prepareForTransport* and *completedTransport* methods have been overridden to save and restore the engine status. If future versions of Concordia solve the serialization problem, then the manual serialization process can be eliminated.

10. Conclusions and Future Directions

In this project, the applications of software agents have been explored. The flexibility of the mobile agent technique makes it quite unique as compared to other techniques. Two application scenarios have been formulated. The first one is the maintenance scheduling between an ISO and a GENCO. The second one is about distributed information retrieval for circuit breaker maintenance that has been integrated with a separate Circuit Breaker Monitoring project. The two applications are mainly exploring the functionality of mobile agents. Two papers have been published for those two applications [23, 24]. A new agent-oriented approach has been studied to fully utilize the benefits of software agents. An example for managing the maintenance workflow has been given [25].

The agent-oriented approach may also be pursued as a future research direction. A single software agent can be further enhanced with intelligence for reasoning and planning, and providing a friendlier user-interface.

Multiple agents can collaborate to form Multiple Agent Systems (MAS). The Motivations for having Multiple Agent Systems (MAS) are:

- To solve problems that are too large for a centralized single agent to do due to resource limitations or to the reliability risks of having one centralized system
- To allow for the interconnecting and interoperating of multiple legacy systems, such as expert systems and decision support systems
- To provide solutions for inherently distributed problems, such as distributed sensor networks or air-traffic control
- To enhance modularity, speed (due to parallelism), reliability (due to redundancy), flexibility and reusability at the knowledge level (shareable resources).

It is believed that intelligent behavior will somehow “emerge” from the interactions of many simple agents, even if the agents are unsophisticated “reactive” entities that behave in a stimulus-response manner using a very simple processing and control mechanism. For this reason, Multiple Agent Systems (MAS) have become the focus of artificial intelligence.

Applications of the agent-oriented approach and MAS will be future research directions.

References

[1] Concordia white paper, http://www.concordiaagents.com/MobileAgentsWhitePaper.pdf.

[2] J. Moubray, "Reliability-centered Maintenance", 2nd edition, Industrial Press Inc. 1997.

[3] Risk and Probability Applications Subcommittee, "The Present Status of Maintenance Strategies and the Impact of Maintenance on Reliability", IEEE Transactions on Power Systems, Vol. 16, No. 4, November 2001.

[4] M. Shahidehpour, M. Marwali, "Maintenance Scheduling In Restructured Power Systems", Kluwer Academic Publishers, 2000.

[5] H. S. Nwana, "Software Agents: An Overview", Knowledge Engineering Review, vol. 11, no. 2, 1995.

[6] Java™ 2 Platform, Micro Edition, http://java.sun.com/j2me/.

[7] JDBC specification, http://java.sun.com/.

[8] Java RMI specification, http://java.sun.com/.

[9] Jess website, http://herzberg.ca.sandia.gov/jess.

[10] CLIPS website, http://www.ghg.net/clips/CLIPS.html.

[11] FIPA web site, http://www.fipa.org/.

[12] Andrew S. Tanenbaum, Maarten van Steen, "Distributed Systems, Principles and Paradigms", Prentice Hall, 2002.

[13] CORBA specifications, http://www.omg.org/.

[14] JmatLink website, http://www.held-mueller.de/JMatLink/.

[15] CPLEX User's Manual, ILog, March 1999.

[16] OmniORB website, http://omniorb.sourceforge.net/.

[17] JacORB website, http://www.jacorb.org/.

[18] M. Kezunovic, C. Nail. Z. Ren, D.R. Sevcik, J. Lucey, W. Cook, E. Koch, "Automated Circuit Breaker Monitoring," IEEE PES Summer Meeting Chicago, July 2002.

[19] S. Russell, P. Norvig, "Artificial Intelligence, A Modern Approach", Prentice Hall, 1995.

[20] Workflow management web resources, http://www6.informatik.uni-erlangen.de/research/wf_ref erences.html.

[21] Grady Booch, “Object-Oriented Analysis and Design with Applications”, 2nd edition, Addison-Wesley, 1994.

[22] Microsoft Agent home page, http://www.microsoft.com/msagent/devdownloads.htm.

[23] X. Xu, M. Kezunovic, "Mobile Agent Software Applied in Maintenance Scheduling", NAPS 2001, College Station, 2001.

[24] M. Kezunovic, X. Xu, D. Wong, "Improving Circuit Breaker Maintenance Management Tasks by Applying Mobile Agent Software Technology", IEEE T&D Conference 2002, Asia Pacific, October 2002.

[25] X. Xu, M. Kezunovic, D. Wong, "Agent-Oriented Approach to Work Order Management for Circuit Breaker Maintenance", IEEE PES Summer Meeting 2002, Chicago, July 2002.