Agent-Oriented Approach to Work Order Management for Circuit Breaker Maintenance

X. Xu, *Student Member, IEEE,* M. Kezunovic, *Fellow, IEEE, and D. Wong, Member, IEEE*

***Abstract*—Compared with Object-Oriented Programming, Agent-Oriented Programming provides higher level abstraction and encapsulation. The software agents are autonomous entities with abilities to adapt the changing environments. An example of carrying out a circuit breaker maintenance work order has been used to illustrate the flexibility and advantages of Agent-Oriented Programming.**

***Index Terms*--Circuit breaker, maintenance, monitoring, agent, power systems, substations, distributed processing.**

I. Introduction

CIRCUIT breakers play a crucial role in power network switching aimed at facilitating both the routine network operation as well as protection of other devices in power systems. To ensure circuit breakers are in healthy condition, periodical inspection and preventive maintenance are performed. Maintenance crews typically carry out the tasks of inspecting and testing circuit breakers manually after a work order is issued. The decision about the time intervals for the testing and inspection, as well as the details related to the maintenance and repair tasks are based on some standards [1-3] or obtained using a maintenance management solution such as Reliability Center Maintenance (RCM) software [4].

Prior to issuing the work order, and during the maintenance or repair work, the information distributed across the utility and stored using different data formats may have to be used. Once the tasks are done, there is a need to file reports and log the maintenance activities. Due to the heterogeneous and distributed property of the information sources, the current information exchange methods involved in the maintenance management practices still require human interference and are thus paper-based. This information exchange method may take both the maintenance crew and the management personnel a considerable amount of time. A new method that can facilitate the whole process of searching the information needed for issuing a work order, verifying availability of the spare parts, and scheduling the work as well as automating the reporting and logging processes after the work is done is desired. The new method should be capable of handling heterogeneous information sources and providing security support for working on public networks. It should also have more flexible structure and more intelligence than the software solutions usually used, therefore it can achieve the functions that the current techniques cannot.

Software agent technique provides a flexible framework for distributed applications. Software agents can travel securely through the internet/intranet to the computers where the information is located. Software agents are self-upgradeable, therefore it is relatively easy to implement applications and have different types of workflow based on the same set of information sources. Also, it is natural to add certain intelligence into software agents and therefore they can act on behalf of the user in some cases. With features to support distributed processing, multi-agent collaboration, intelligence for reasoning and planning, and more friendly user-interface, software agents may be a good candidate for the next generation information exchange and management systems.

In this paper, we considered how software agents might be applied in the scenarios of circuit breaker maintenance from the viewpoint of facilitating the tasks of the management personnel and the maintenance crews. Different from our previous work [5,6], the new approach is totally agent-oriented. This new point of view may provide an even more flexible software structure and more user-friendly interfaces. At first, the problems of circuit breaker maintenance are reviewed, following by a brief introduction to software agent technique. After that, the new agent-oriented approach is discussed and a work order agent is implemented as an example. The benefits are summarized at last.

II. Circuit Breaker Maintenance Tasks

The circuit breakers consist of the interrupter assembly (contacts, arc interrupters and arc chutes), operating mechanism, operation rod, control panel, sealing system, and breaking medium (SF6, oil, vacuum and air). To ensure the performance of a circuit breaker, all the components should be kept in good condition, therefore time-directed preventive maintenance has been widely adopted. The preventive maintenance tasks include periodic inspection, test, and replacement of worn or defective components and lubrication of the mechanical parts. The maintenance intervals are usually determined using experiences or following the recommended

This work was supported in part by PSerc consortium, an NSF I/UCRC, and in part by Texas A&M University.

X. Xu is with Texas A&M University, College Station, TX 77843 USA (e-mail: xuxj@ee.tamu.edu).

M. Kezunovic is with Texas A&M University, College Station, TX 77843 USA (e-mail: kezunov@ee.tamu.edu).

D. Wong is with Mitsubishi Electric Research Laboratories, MA 02139 USA (e-mail: wong@merl.com).

schedules provided by the vendor or standard [1].

The maintenance practices can be divided into three categories: corrective maintenance, preventive maintenance, and predictive maintenance [4]. The different strategies are summarized in Table I. Each maintenance strategy has its own advantages and disadvantages, and thus most suitable application scenarios. A systematic solution is to utilize the Reliability Centered Maintenance (RCM) methodology. It performs analysis of the failure modes and the cause-effect impacts on the devices as it tries to find which strategy is the most cost-effective and appropriate for an application. The result of utilizing the RCM techniques and tools will be an optimal maintenance schedule for a specific application scenario.

TABLE I
MAINTENANCE STRATEGIES

Strategy	Description
Run-to-failure maintenance (Corrective, repair only)	The repair and restoration of equipment or components that have failed or are malfunctioning and are not performing their intended function
Time-directed maintenance (Preventive)	The periodic and planned maintenance actions taken to maintain a piece of equipment within the expected operating condition. It extends the equipment life and is performed prior to equipment failure to prevent it. This includes technical specification surveillance, in-service inspection, and other regulatory forms of preventive maintenance
Condition-directed maintenance (Predictive)	The continuous or periodic monitoring and diagnosis in order to forecast component degradation so that as-needed planned maintenance can be performed prior to equipment failure. Not all equipment conditions and failure modes can be monitored; therefore, predictive maintenance must be selectively applied.

The location of the information needed to perform maintenance can be the enterprise maintenance system, the substation data concentrators and the maintenance crew's computer.

The information about the spare parts, test procedures, historical maintenance records, and instruction manuals, etc. is typically accessible in the enterprise maintenance system. Also, 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 what devices.

The information about the substation equipment may be retrieved from the substation computers or data concentrators. With the introduction of continuous monitoring of circuit breakers, the real-time data becomes available in the substation concentrators for further accessing. The continuous monitoring instrument may measure the coil current profiles and timing of the circuit breaker switching 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. Also, 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 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 (encrypted data transmission, user authentication and authorization)
- Efficient network bandwidth usage
- Robust and fault-tolerant communication over unreliable environment and portable personal communication devices
- Ability to integrate with heterogeneous systems
- Automatic software update to ease the user burden

During the whole maintenance workflow, from issuing the work order to completing the work order and filing reports, some tasks such as checking the availability of circuit breaker parts, recording the current circuit breaker status, etc. can be automated. Those kinds of works should be taken care of by the software to save the time of the maintenance crew and prevent human errors. Also, the software should be adaptive to the environment. For example, when certain parts are not available, the software should be able to change the original plan. Those preferences imposed on the software will require the software having certain intelligence.

The maintenance management systems and the maintenance crews may prefer some higher-level aid. The software should have:

- Ability to check the feasibility of a certain task
- Ability to plan the next step
- Ability to adapt to changing environments

III. AGENT-ORIENTED PROGRAMMING

A software agent is defined as "an autonomous process capable of reacting to, and initiating changes in, its environment, possibly in collaboration with users and other agents" [7]. A typical software agent is shown in Fig.1. The agent reacts on the input percepts and may create output actions. The percepts will come from sensors or other agents or systems. The actions may be used to drive some effectors or communicate with other agents or systems. The program will control the agent according to the internal status. The knowledge and algorithms of the agent will affect its decision process. Better algorithms and more knowledge usually will give an agent better chance when competing with other agents.

Software agents may also be able to provide friendlier user-interface [8]. How to implement an efficient user-machine interface is not an easy problem. Agent techniques may

provide a promising solution.

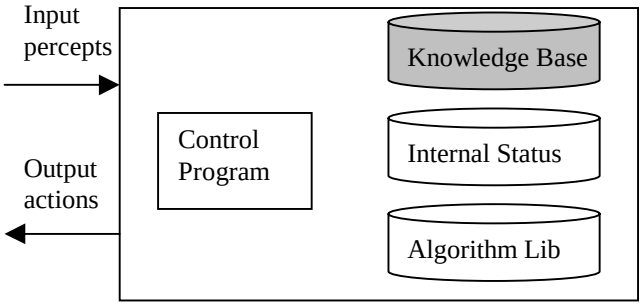

Fig. 1. A typical software agent

Compared with objects, one basic characteristic of software agents is autonomy, which means agents can exist independently. This autonomy usually requires the agents to act reactive or proactive and thus have abilities for reasoning, planning, or even learning. Software agents provide higher level abstraction and encapsulation, and they are generally more intelligent and more adaptive.

While objects are the basic modeling units in Object-Oriented Programming (OOP), software agents will be exclusively used to model the physical world in Agent-Oriented Programming. The basic idea of OOP is to encapsulate the implementation details inside objects, thus achieving more flexible software structure. Unlike in OOP, where the objects only provide services and are usually passive, the agents are autonomous and usually active. Agents may initiate actions when necessary and often possess certain intelligence.

For example, in the cases of circuit breaker maintenance, we can represent a Work Order (WO) as an object. The Work Order (WO) object may expose several methods to update its status, log final results, etc. An outside process will be in charge of creating, manipulating and destroying the WO object. Usually the outside process will model the workflow of carrying out a work order. The reasons that we do not build the workflow into the WO object are: (1) In general, objects are passive and only act as servants. (2) The whole workflow may involve many other objects and must be adaptive to the environment. It is difficult to achieve this by OOP. If utilizing Agent-Oriented Programming, a Work Order (WO) agent can be used to model not only the description of the work order but also the procedures to carry out the order. The procedures can be described using rules or predicate logic and will be interpreted at runtime. Since the rules or logic can be updated dynamically, the agents can adapt to new procedures and situations easier. No additional steps of compilation and deployment are required.

It is possible to implement software agents using any programming language (e.g. C/C++, Java). For example, we can create an agent using C++ language and embed a copy of CLIPS engine inside the agent. The workflow will be described using IF-THEN rules, and the agent will find out what is the next action based on the rules and the current status. Although we can use any language to build software agents, some languages are more suitable than others. Java language is such a language due to its cross-platform support and standard programming libraries. Several software agent systems, which are built using Java, are Concordia [12], Voyager [14], and JADE [15].

There are still some problems in Agent-Oriented Programming. First, there is no universal format for knowledge representation. Usually the choice of the knowledge representation is domain specific. Agents from different domains will have difficulties to exchange knowledge. Second, software agents built using different agent system may not be able to communicate with each other due to the different communication protocol or message formats. There are some efforts of standardization under going. For example, the Knowledge Query and Manipulation Language (KQML) has been proposed for exchanging information and knowledge among agents [9]. And the Foundation for Intelligent Physical Agents (FIPA) was formed in Europe in 1996 aiming at producing standards for the interoperation of heterogeneous software agents [10]. Several implementations compliant to the FIPA specification are JADE [15], FIPA-OS [16] and Zeus [17]. Also, the newly proposed Java API for agents – JSR87 [11] is based on FIPA and will supersede FIPA. Third, unlike many tools in existence for supporting Object-Oriented Programming, the development tools for Agent-Oriented programming are almost non-existent.

IV. Application Example

The application scenario is shown in Fig. 2. Three computers are used to represent the enterprise maintenance system, the substation concentrator and the maintenance crew respectively.

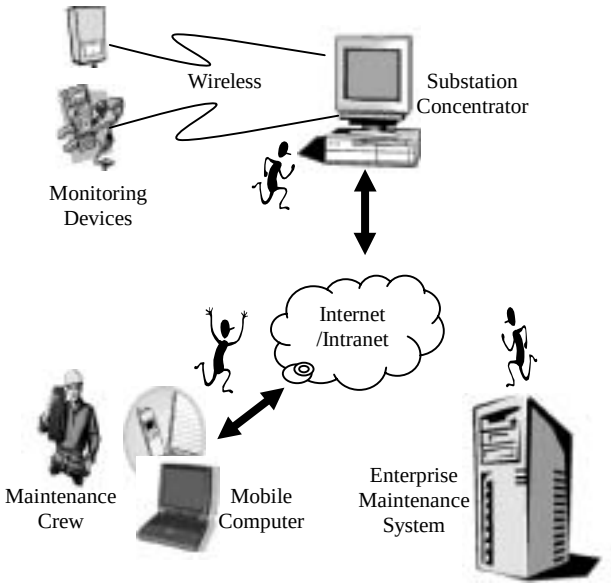

Fig. 2. Application scenario

The enterprise maintenance system may contain the maintenance history database, the RCM system, warehouse inventory system, and other information. The substation

concentrator is in charge of collecting data from the sensors installed on the circuit breakers. Some analysis software may be running and a status report describing the circuit breaker operation can be generated. The maintenance crew uses a mobile computer to access the information and prepare report utilizing mobile software agents.

In our previous work [6], the mobile agents are mainly used to address the problems of a distributed, heterogeneous information retrieval. A prototype system with a Graphic User Interface has been built to demonstrate the advantages. The primary advantages can be summarized as (1) low network bandwidth consumption and unlimited system extensibility. (2) support of multiple platforms, and (3) reliable and secure transmission.

Merely viewing agents as another type of information retrieval tools does not utilize all their potentials. For example, the users are still supposed to drive the maintenance workflow to fulfill a work order. They have to decide when to retrieve what information or what is the next place to go. Also they will usually need to update the plans according to the changed situations. For an inexperienced worker, who is not yet very familiar with the whole workflow, a software assistant that can aid the whole process is desired.

We have utilized Agent-Oriented Programming to re-implement some of the circuit breaker maintenance scenarios. A work order agent is used to represent the actual maintenance task. Once created, the work order agent will be in charger of driving the whole maintenance workflow util it is finished. The maintenance crew no longer needs to plan for the steps to fulfill a work order.

Concordia [12] is selected as the supporting mobile agent platform. Jess [13], a Java implementation of CLIPS, is chosen as the embedded inference engine. The evaluation versions of both Concordia and Jess can be downloaded from the Internet. Jess is free for non-commercial use. We also have implemented a Java interface for CLIPS, which provides another option.

An example of maintenance workflow is given in Fig. 3.

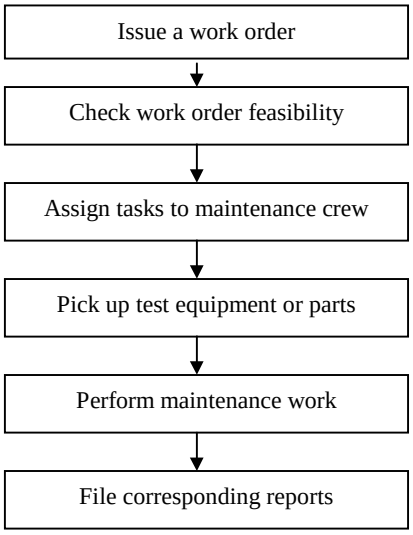

Fig. 3. An example of maintenance workflow

Usually, the work order is issued by a maintenance management system. It 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 some parts, then those parts must be available. Otherwise, the work order must be pending and waiting for the parts arriving. Then the work order will be dispatched to an available maintenance crew. After receiving the order, the crew may need to pick up some testing equipment or parts before going to the substation to perform the actual maintenance work. After the maintenance work is done, some reports may be required.

We use IF-THEN rules to model the workflow of Fig. 3. 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. 4.

```
(defrule rule_task_assignment
  (maint_task ?task $?param)
  (feasible ?task)
  (available_crew ?crew)
  =>
  (printout t "assigning " ?task " to " ?crew crlf)
  (agent_func assignTask ?task)
)
```

Fig. 4. 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 events of interest (e.g. new-parts-arrived, 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. 5. 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.

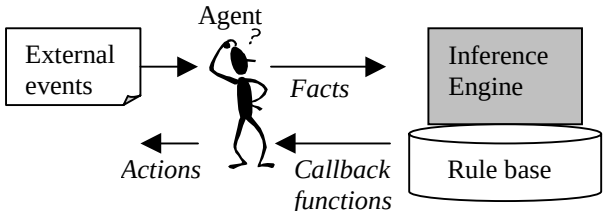

Fig. 5. Interaction between rules and agent

A. Issuing a work order

The enterprise maintenance system will create a work order agent for dispatching a work order. The work order 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 a “sleep” status can begin to enter the “ready” status.

In an ideal case, the maintenance crew should have a personal agent, which is in charge of arranging its schedules. And 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. Also the agent will record the name of the crew. 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 to finish the given work order. Therefore, the agent will be in charge of driving the maintenance workflow.

First, according to the type of the 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 order of the warehouses is decided according to the distance to the circuit breaker and 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 the inability to push forward the workflow. When the parts arrive, the agent will receive the notification and it 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. Also, 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 to prepare reports. Usually, the formats of the reports are standardized in a company and the agent can automatically fill most of the information in the reports.

The work order agent will return to the enterprise maintenance system and submit the report. Also, 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.

V. CONCLUSIONS

Agent-Oriented Programming has been used to implement a software agent capable of assisting the maintenance crew in performing circuit breaker maintenance. The agent uses rules to model the maintenance workflow where the workflow is driven by events.

The agent-oriented approach provides more flexible software structure by higher level of abstraction and encapsulation. Since the modeling is much closer to the real world concepts, the software becomes easier to understand.

Software agents are more adaptive to the changing environments. By subscribing to the events of interest, the agents can be notified timely and act accordingly. The rules are interpreted by the embedded inference engine and can be updated and changed dynamically.

The agent-oriented approach can provide higher level aid to the maintenance crew. By integrating artificial intelligence, agents can guide the crew during the whole maintenance workflow.

VI. REFERENCES

[1] United States Department of the Interior Bureau of Reclamation, “Maintenance of Power Circuit Breakers”, Internet Version, 1999, [Online]. Available: http://www.usbr.gov/power/data/fist_pub.htm

[2] *IEEE Guide for Diagnostics and Failure Investigation of Power Circuit Breakers*, IEEE Standard C37.10-1995, Dec. 1995.

[3] *IEEE Guide for Selection of Monitoring for Circuit Breaker*, IEEE Std C37.10.1-2000, 2001

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

[5] X. Xu and M. Kezunovic, "Mobile Agent Software Applied in Maintenance Scheduling," in *Proc. 2001 North American Power Symposium*, pp. 497-503.

[6] M. Kezunovic, X. Xu, and D. Wong, "Improving Circuit Breaker Maintenance Management Tasks by Applying Mobile Agent Software Technology", IEEE/PES T&D Asia Conference, Yokohama, Japan, October 2002

[7] A. Tanenbaum, and M Steen, *Distributed Systems, Principles and Paradigms*, New Jersey: Prentice Hall, 2002 p173

[8] Collagen website, http://www.merl.com/projects/collagen/

[9] KQML Specification, http://www.cs.umbc.edu/kqml/

[10] FIPA website, http://www.fipa.org/

[11] Java Specification Request (JSR) 87, http://jcp.org/jsr/detail/87.jsp

[12] D. Wong, *et. al*., "Concordia: An Infrastructure for Collaborating Mobile Agents", First International Workshop: MA '97, April 7-9, 1997, Lecture Notes in Computer Science, Vol 1219, Springer-Verlag, Berlin, 1997, pp. 86-97.

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

[14] Voyager website, http://www.objectspace.com/
[15] JADE website, http://sharon.cselt.it/projects/jade/
[16] FIPA-OS website, http://fipa-os.sourceforge.net/
[17] Zeus website, http://www.labs.bt.com/projects/agents/zeus/
[18] A. Tveit, "A survey of agent-oriented software engineering," in *Proc. of The First NTNU CSGS Conference, Trondheim, Norway, May 2001*.

VII. BIOGRAPHIES

Xiangjun Xu (S'99) received his B.E and M.E. degrees from Southeast University and Shanghai Jiaotong University, all in electrical engineering, in 1992 and 1995 respectively. After that, he worked as a lecturer/researcher in Shanghai Jiaotong University. Since Sep. 1998, he has been with Texas A&M University pursuing his Ph.D. degree. His research interests are computer application on power systems, signal processing, artificial intelligence, and fault analysis.

Mladen Kezunovic (S'77, M'80, SM'85, F'99) received his Dipl. Ing. Degree from the University of Sarajevo, the M.S. and Ph.D. degrees from the University of Kansas, all in electrical engineering, in 1974, 1977 and 1980, respectively. Dr. Kezunovic's industrial experience is with Westinghouse Electric Corporation in the USA, and the Energoinvest Company in Sarajevo. He also worked at the University of Sarajevo. He was a Visiting Associate Professor at Washington State University in 1986-1987. He has been with Texas A&M University since 1987 where he is the Eugene E. Webb Professor and Director of Electric Power and Power Electronics Institute. His main research interests are digital simulators and simulation methods for equipment evaluation and testing as well as application of intelligent methods to control, protection and power quality monitoring. Dr. Kezunovic is a registered professional engineer in Texas, and a Fellow of IEEE.

David Wong (M'92) is currently a Deputy Director at Mitsubishi Electric Research Laboratories. He is an expert in mobile agent technology and the primary evangelist for Mitsubishi's mobile agents initiative. His background also includes substantial work in transactional message queuing systems and distributed transaction processing. Prior to joining MERL in 1994, David worked on the advanced development and performance analysis of transaction processing systems at Digital Equipment Corporation. He has also taught and conducted research at Brown University and the University of Connecticut. David received his B.S. degree in chemical engineering from Brown University and the Ph.D. degree in computer science from the University of Connecticut in 1980 and 1991, respectively.