State-oriented Maintenance of Electrical Power Systems with Dynamic Multi-agent Network

Miroslav Prýmek and Aleš Horák
Faculty of Informatics, Masaryk University
Botanická 68a, 60200 Brno, Czech Republic
{xprymek,hales}@fi.muni.cz

ABSTRACT
This paper deals with the problems of maintenance of electrical power systems (monitoring and control) and argues for the assets of state-oriented maintenance as opposed to the time-oriented maintenance. The reasons for state-oriented maintenance are justified with a detailed description of design and implementation of a system dedicated to simulation of power system processes based on the multi-agent approach. The resulting system consists of individual agents that communicate by means of a heterogeneous dynamic network. The prototype implementation of the system is based on open standards CORBA and KQML that are also briefly described in the text.

Key Words: electrical power system; multi-agent dynamic network

1. Introduction
Enterprises running the power systems are making a big effort to make the reliability assurance process more flexible and adequate to the real conditions of the power facilities. Simple time-based controls of the facilities do not take into account the rate between the maintenance costs and the money loss caused by the failures in the energy delivering process. Hopefully the more appropriate way of dealing with the reliability assurance will be the new methodology of maintenance based on the real state of the facility and the amount of the energy delivered (or “potentially undelivered”) by the facility.
To achieve more reliable power delivering many of the facilities are backed up. Routine action after failure is to change the active lines topology or to switch to a backup facility to restore power delivering. Each failure is precisely recorded into a failure database (SCADA). The process of the power system maintenance and management, failure recognition and function restoration can be optimized at least from the economic point of view. The problem is somewhat classical: to minimize the overall costs (maintenance, restoration, undelivered energy) which is a function of the facilities condition and its features, dynamic network topology and also the restoration process methodology. The problem is that the function is mainly uncomputable and it is not even clear which variables are to be measured to figure out the function.
Our work has two main motivations: (1) To support the development of the condition-oriented facility maintenance methodology by the information technologies. (2) To develop a software for simulation of the various features of the power system facilities. The task is concentrated mostly on simulation of the electrical energy flows, outages due to facility failures, and their impact on the overall sum of the non-delivered energy.
In its first stage, the simulator is meant to be a research tool extending classical tools for reliability assurance like SCADA databases and statistical tools for failure-data exploration. There is a narrow connection between the exploration of the data in SCADA databases and developing the simulator.
2. The Rice System
Our software framework, called “Rice” is an experimental implementation of the power system simulator, which implements all of the above-described features needed for developing the new reliability assurance methodology. It is based on these main principles, described further: (1) it is a multi-agent system, (2) it can be used to research behavior of the dynamically changing environment, (3) as to practical implementation, it is decentralized, modular and open. This is necessary especially for use in geographically dislocated areas (e.g. online facilities monitoring).

2.1 Multi-agent Approach
Rice is based on the principles of the multi-agent approach (see the Figure 1). Each facility of the power system is supposed to be autonomous (self-deciding and autonomously reacting to system events) and so represented by one software agent.

For general introduction to agent-based software engineering methods, see i.e. [1]. Practical examples (including case studies) of an agent-based architecture usage in the industrial control systems are presented in [2].

Key idea of the multi-agent system is the idea of the agent itself. Hence there is plenty of definitions what agent is or should be, we can stress some common thoughts about agents:

1. Agent is cognitive. It watches its surroundings and makes its own “cognitive map” of the world. All events taking place in the agent's environment are filtered by the agent according to their importance for agent's aim and important ones are stored in agent's knowledge base (memory).

2. Agent is autonomous. There is no prepared plan of actions agent should do to achieve his aim. In every time slice agent itself chooses the best plan according to his knowledge base.

3. Agent is communicative. It can ask any other agent for any information he needs to fill his knowledge base. Communication patterns (i.e. topology of the communication net) are strictly dynamic. The structure is built up ad-hoc according to agents' needs for some knowledge. There are no “hardwired” communication paths.

4. Agent is social. Main features of the multi-agent systems are functions of the system as a whole, not any particular agent.

As we have said in our system every power system facility is represented by one
software agent. Agents interaction standard is defined by the set of messages agent must understand. These message sets are hierarchically arranged so they grow up into the tree of agent types. This principle is similar to the principle of the object-oriented programming (discussed later).

Every agent is therefore viewed as a black box with some inputs and outputs. To fulfill the idea of the multi-agent system, every agent decides on his state only from the state of his inputs. This leads to the local-interactions-only system. And further: no information passing between agents is obligatory. Every information must be obtained by the question passed to the particular agent (one question can lead to the time-based or event-based series of answers). This rule asserts that there is no superfluous flow of information.

The “1:1” design (one facility - one agent) of the multi-agent system is necessary for interaction with SCADA databases, as we will see.

2.2 Dynamic Environment

It is not easy to track events in dynamically changing environment. Because we are in the situation where we do not know precisely what variables are of interest for exploring the facilities failure rates (and thus also for maintenance necessity and cost), things are even harder. One possible approach is to experiment with the auto-adaptive networks.

In our system there is no “hardwired” topology or information flow. Now, we implement simple agent for simulation of the power line. The only thing that it does is that it copies incoming energy flow to his output. In the next stage of the development, we are going to replace this agent with a more sophisticated one which can simulate power distribution. This could demand closer communication with inputs and outputs. This information will be obtained on-demand, when it is needed. This is the dynamicity as to the amount of a communication.

Similarly the network is dynamical as to the topology. The topology is built up by the series of question and answer couples. When the connection between two agents is no more needed (e.g. because the power line connecting them is down), communication is shut down and if there is another agent which can facilitate the same function (and so restore the energy distribution), new connection with it is established. The result is that the system can be used for simulating ever-changing emergent environment and for measuring the efficiency of a transportation system in such an environment.
2.3 Decentralized and open
The application environment is very heterogeneous. It is absolutely necessary to use technologies independent on hardware platform and operating system. The framework also supports wide range of programming languages (due to CORBA). In the contemporary prototype Rice1 Python and Java are used.

We also take into account possible future needs to integrate the proposed system with the contemporary software and hardware solutions of the power system facility vendors. To make this integration easier, the system is based on standard technologies - with open specifications or even implementations.

2.4 Architecture
After the specification of all the system requirements (see [3]), we have decided that the best results will be acquired by implementing the system using standard technologies commonly used in the multi-agent systems area - a combination of two communication protocols, CORBA and KQML.

Usage of the CORBA-KQML solution was published in the literature (e.g. [4], [5] and [6]). The multi-agent approach has been used even in the area of the power systems monitoring and control (e.g. [7], [8] and [9]).

disadvantages and implementation details were introduced in [10]. We will only shortly review it here. A schema of the communication system is displayed in the Figure 2.

3. Implementation
The first layer, TCP/IP is the standard Internet communication protocol allowing the system to consist of geographically dissipated components. The idea is that some of the simulating agents could be in future replaced by the on-line sensors monitoring the real facility operation. This really breaking change is seamless and does not demand deep code intervention because of the multi-agent architecture and the KQML flexibility.

The second layer, CORBA (Common Object Request Architecture) is a standard architecture for inter-process (and even inter-machine) procedure calling, used widely in the system integration solutions and decentralized software. It is strictly object-oriented. The object's API are defined in an abstract Interface Definition Language (IDL) which can be translated into many real programming languages. Thus it is simple to integrate code pieces written in different languages and running on different operating systems and machines.
Third layer, KQML (Knowledge Query and Manipulation Language) is a language developed specially for usage in the multi-agent systems. It is an abstract definition of the message purpose, not the message content itself. This way, the meaning of the message and its object is separated which is very useful for system integration and definition of abstract message types. (This subject is little bit complicated and it is discussed in [10]).

The last but not least layer is the message object itself. Thanks to KQML structure, it can be defined in many languages and on-demand structured. The language used can even vary between same messages and can be dynamically negotiated by the communicating agents.

### 3.1 With OOP Principles

CORBA translates all thru-network calls to programming language concepts similar to calls to local objects. This principle is very useful when designing particular agents. We can use standard OOP programming principles.

Because our message API, as said, is also hierarchically structured like standard OOP inheritance, the whole system is OOP designed. This leads to very simple implementation of agents with new features.

E.g. think of agent simulating simple time-based failures (frequency of failures can be determined from real-life SCADA entries). Because all (relatively complicated) functionality related to message passing is defined in common agent definition, implementation of such an agent is a matter of about ten lines of code (in python).

All events in the system are performed only by message passing. For instance an outage on the line is performed by message “line is at zero V”. Receiving agent decides on his own, how to react to this type of a message. An agent with a backup power supply can just switch to his UPS (see the Figure 3a).
Agent asked his inputs to send him all notifications about their status changes (in the language of KQML he “subscribes” itself to this kind of information). He also demanded that this kind of messages should me marked in a field “inReplyTo” with the value “inputStatus”. His parent classes automatically deliver messages about some agent's status change to the method rLAgentStatus, so the agent has to reimplement this method. Messages about input status changes are filtered according to the value of “inReplyTo”.

If we had simpler agent “fault tolerant power line” which copies its input to output, this method is displayed in the Figure 3b). We can see that agent sets its own status according to the status of the input. Again, parent classes automatically inform all potential “subscribers to this agent's status” that the status has changed.

On these very simple illustrating examples, we can see how OOP principles can be adopted in the de-facto non-OOP field of KQML messages.

4. Exploitation of Failure Database

More interesting is usage of Rice in association with classical SCADA databases. These databases are primarily used to determine particular facility failure rates, various conditions of the failures, defective facility series etc. Such information is processed by means of the classical data mining and statistical analysis. In spite of that data mining is relatively advanced field, it can be hard or impossible to harvest some types of information from the database, which entries map only particular failures, outages, etc.

But SCADA can be used for “profiling” particular facility type and its behavior in the network which can be then used in development of an agent simulating this facility. The particular methodology of this process must be defined yet and it is our plan for the near future.

We can also use backward process. In Rice, agents send information about their state to the application which visualizes processes taking place in the network. The same principle can be used to create simSCADA – simulated SCADA. By comparing data in the real SCADA and simSCADA, we can numerically evaluate simulation accuracy.

Moreover, because failures are simulated by message passing, it is relatively simple to track down failure origin and join this original failure with its consequences (measure precisely the amount of undelivered energy). This could be hard with classical SCADA and we consider this a promising feature of our system. It is simple to measure the overall sum of the undelivered energy and use this measure for the optimization of the agent behavior principles.

This is promising future of our system. This feature can make Rice an important tool in the power systems simulation and research.

4. Conclusions

We have presented a new prepared system for state-oriented maintenance of electrical power systems. The system starts with the design of its flexible communication framework that allows adding new functionality for intelligent failure monitoring and prevention in a versatile and standard way.

Currently, the system exploits the valuable databases of power system failures which have been constructed for several years in 9 region of Czech Republic and Slovakia. Further development aims at integrating the results in the stochastic simulation of failures of particular network elements which, as we hope, will be an important step towards the tools and systems helping to effectively control the complex behavior of such emergent systems.

Acknowledgements

This work has been partly supported by Grant Agency of the Academy of Sciences of CR under the project T100300414 “Intelligent methods for increasing of reliability of electrical networks.”
References:


