Power Distribution Network Reliability Modeling Based on the Statistical Data and Multi-agent Simulation

Miroslav Prýmek1), Aleš Horák1), Radomír Goňo2), Stanislav Rusek2)

1) Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {xprymek,hales}@fi.muni.cz

2) Faculty of Electrical Engineering and Computer Science
VSB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava – Poruba, Czech Republic
E-mail: {radomir.gono,stanislav.rusek}@vsb.cz

ABSTRACT

The paper describes an innovative approach to the exploration of the power distribution network reliability problem. The approach is based on dynamic modeling of the power distribution network features, its failures and their consequences. The actual modeling processes are conducted by a network of independent and autonomous software agents whose behavior is defined by a declarative language. A particular network model can be machine-generated on the basis of a real-life failure database data. The hypothesis is that we can bring new findings to the research field by joining the real-life statistical data with the highly flexible multi-agent simulation.

Keywords – reliability; power distribution network modeling; multi-agent system

1 INTRODUCTION

Institutional changes taking place all over the world change drastically the approach to power supply reliability. It develops towards a purely commercial matter between suppliers and their customers. The supply that does not comply with agreed qualitative parameters will lead to trade disputes and financial settlements. So-called undelivered energy, including its valuation, arrives on the scene.

The world centers of reliability analyses provide electronic databases of information about availability of electronic and no electronic components and distribution function of kind of failure. Unfortunately, databases do not contain data about electric power equipments operated in our conditions.

To determine the reliability of network elements and the supply of electrical energy to consumers, it is necessary to observe failures and outages in the transmission and distribution of electrical energy. The calculation of probably unsupplied energy is only possible just on the basis of results of reliability calculations.

What is of great importance is the connection of databases of regional distribution companies to the overall database that will be continuously supplemented and broadened. In this way the statistically significant set of data will be created that will better and more quickly describe the real condition of network equipment. For this reason, data necessary for the analysis of component reliability is processed centrally at the research workplace at the VŠB - Technical University of Ostrava, Czech Republic. However, it is also necessary to unify the processing of data on failures and outages in particular companies so that reliability results could be compared.

1.1 History of Failure Monitoring

Calculation of reliability of equipment started to be use in the area of military engineering. In this area the theory of reliability computations is best elaborated. A lot of reliability theories arose during guided missiles development.

The Research of electric power system reliability started in the 40th of last century in the U.S.A., later in the USSR and Great Britain. From the 50th the research of electric power system reliability has carried on in all developed countries.

In the former Czechoslovakia electric power system reliability started to be discussed even during 60th of last century. The turning point in monitoring of the reliability was 1974 when the Regulations 2/74 for electric power systems CEZ and SEP were released. These regulations unified monitoring of failures, outages and damaged equipments for all distribution companies in Czechoslovakia. Since 1975 exclusive database of failures and outages began to rise.

Technical aspects of this database were corresponding to the period of its creation; workers manually filled in particular items into the uniform forms according to uniform codebook. Then total database was saved in the form of text files. Of course, it made more difficult to work with database, evaluating of inquiries was slow and a lot of processing (e.g. mean failure duration evaluation) had to be made manually from separate printing groups. Nevertheless, this database was (and is) very valuable background for reliability determination. Unfortunately, the database filling has been stopped since 1990 because of political and social changes. Separate distribution companies obtained independence and they have started introducing their own systems for monitoring of reliability since the 90th of last century. Total database was not form further.

Specialist group of ČK CIRED started to deal with problems of reliability. First calls for reintegrating databases of failures and electric power supply outages from REAS were already claimed at the first meeting of this group in 1997 where the questions about methods of reliability calculations were discussed. At the meeting of REAS in 1999 there were decided that they would uniformly monitor global indices of reliability and also reliability of chosen pieces of equipment and that data necessary for analyses of item reliability and global indices would be stored centrally at the research workplace of the Department of Electrical Power Engineering VŠB - TU Ostrava where item reliability would be also processed. Data would be hand on and process since the year 2000.

1.2 Reliability Calculations

The majority of reliability calculations are performed in the following way. On the basis of the knowledge of reliabilities of particular elements of the system, the calculation of total system reliability is executed. That is why the reliability calculations have two basic phases. The first phase represents input reliability data acquisition, the second phase then reliability calculation itself. The next phase may be the evaluation of computing results, or the determination of reliability increasing measures.

In virtue of experience it is necessary to state that in most cases, the acquisition of reliable input reliability data is far more complicated and labor-intensive than the performance of reliability calculation itself.

1.2.1 Main Objectives of Monitoring

The principal goal of monitoring events occurring in the distribution networks is to secure the reliable supply of electrical energy to consumers in accordance with the Distribution Network Grid Code [2]. The rate of reliability may be determined from databases of events by means of global indices of supply reliability or reliability indices of particular elements.

1.2.2 System-Oriented Data

The majority of utilities create statistics for the reliability of network components, including lines, transformers, etc. They are especially collected to identify an unreliable piece of equipment and to be used as the input into probability calculations of system behavior.

Basic data on the reliability of equipment and elements of systems is as follows:

- Failure rates of particular pieces of equipment and elements.
- Outages of the piece of equipment due to maintenance and inspections.
- Outages of the piece of equipment due to operating works on the piece itself and labor safety securing in the vicinity of live parts of the distribution system.

This data serves to evaluate properties of pieces of equipment already operated (or a piece of equipment of the certain type of the selected supplier), to select new pieces of equipment, to assess the time suitable for restoring pieces of equipment at the end of their lives, to choose the operating mode of the HV network node, and others.

1.2.3 Customer-Oriented Data

This data is processed in the statistics concentrated on customers' ideas about supply quality in a form of quality indices oriented towards customers' needs.

The reliability of supply to the given customer may be usually assessed by the following so-called “global indices”:

- The number of interruptions (outage number/year/-customer).
- The total duration of all interruptions (min/year/-customer).
- The average duration of one interruption (min/outage).

These indices recommended for this purpose by UNIPEDE characterize the mean average reliability of supply and its consequences from the customer's point of view. They are usable primarily in relation to consulting firms, governing bodies and comparison between particular REAS. Basic data on connections of new customers represents a special utilization; indices being related to the points of supply.

However, in relation to common customers the limits, within which these indices move, and the distribution of their rates are important.

Failure rates of certain components tend to vary with age [1], [2]. A graph that is commonly used to represent how a component's failure rate changes with time is the bathtub curve [3]. Many parameters in the field of reliability vary from component to component or from situation to situation. Random variables are represented by probability distribution functions [4], [5], [6], [7].

Failure rates of overhead distribution equipment are, in general, very system specific due to their dependence on geography, weather, animals and other factors [8].

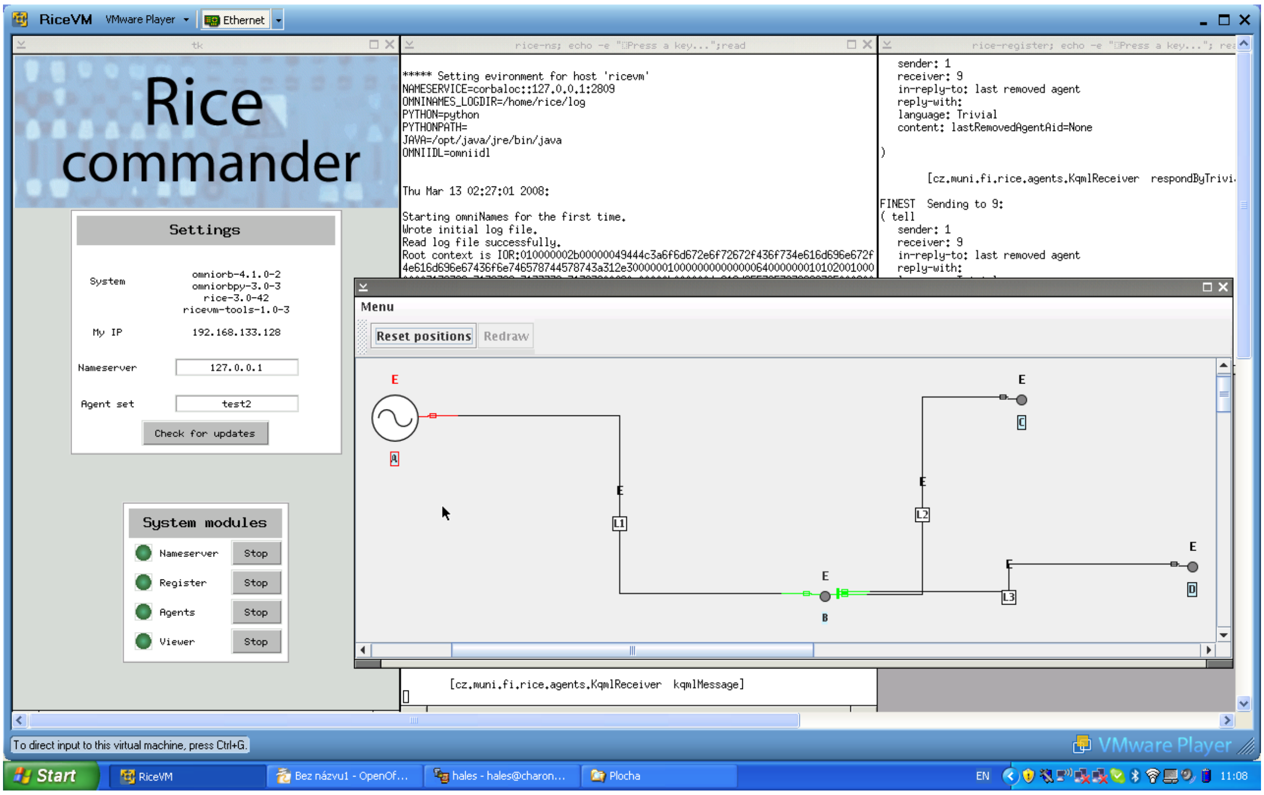
2 MULTI-AGENT SYSTEMS

As opposed to the classical software design which is based on the monolithic or at least central-directed approach which can easily reach its boundaries as to the complexity of the simulated problem or resource usage effectiveness (scalability), multi-agent systems approach is based on the ground-up design whose main idea is to focus on the functions of the fundamental system components perceived as black boxes and defined by their *behaviour* rather than their data and code structure. In this point of view, multi-agent systems (MAS) can be seen as a new form of the objective-oriented software design. With such a design, software system can be easily decomposed and decentralized which leads to better scalability and maintainability.

Every component of the MAS (an "agent") is strictly *autonomous*. The agent takes actions and makes decisions based solely on its knowledge of the surrounding world. This eliminates the requirement of a central management and data-storing unit which can easily form a bottleneck of a system. On the contrary, in the MAS there are only *local interactions* between agents.

Another difference is a mean of the data flow and information (knowledge) interchange in the system. In the traditional software design, there must be a preliminary (central) plan of what data should be interchanged between the software entities. This data are sent at the "discretion" of the data-producing/owning entity. This we can call a "push strategy". On the other hand, communication in a MAS is based on the decision of the agent that some piece of information is really *necessary for his actions* and *therefore* it is acquired from the agent which owns the information. This can be called a "pull strategy".

This data-exchange strategy ensures that only really necessary data is propagated through the agent network and in



System
omniorb-4.1.0-2
omniorbpy-3.0-3
rice-3.0-42
ricevm-tools-1.0-3

My IP: 192.168.133.128

Nameserver: 127.0.0.1

Agent set: test2

```
rice-ns; echo -e "Press a key...";read

***** Setting evironment for host 'ricevm'
NAMESERVICE=corbaloc::127.0.0.1:2809
OMNINAMES_LOGDIR=/home/rice/log
PYTHON=python
PYTHONPATH=
JAVA=/opt/java/jre/bin/java
OMNIIDL=omniidl


Thu Mar 13 02:27:01 2008:

Starting omniNames for the first time.
Wrote initial log file.
Read log file successfully.
Root context is IOR:010000002b00000049444c3a6f6d672e6f72672f436f734e616d696e672f
4e616d696e67436f6e746578744578743a312e300000010000000000000064000000010102001000
```

```
rice-register; echo -e "Press a key..."; rea
   sender: 1
   receiver: 9
   in-reply-to: last removed agent
   reply-with:
   language: Trivial
   content: lastRemovedAgentAid=None

)

        [cz.muni.fi.rice.agents.KqmlReceiver  respondByTrivi

FINEST  Sending to 9:
( tell
   sender: 1
   receiver: 9
   in-reply-to: last removed agent
   reply-with:
```

```
    [cz.muni.fi.rice.agents.KqmlReceiver  kqmlMessage]
```

Fig. 1: Testing Rice installation in a virtual machine environment for easy deployment.

particular makes it possible to simulate *highly complex and dynamically changing environment* because no preliminary knowledge about the data flow is needed and so no such mechanisms have to be hard-coded into the system which would make it less flexible for the new future configurations.

As you can easily see, the crucial thing for the implementation of a problem by using a MAS (aside of the implementation of the MAS communication infrastructure itself) is to define the behavior of the agents. In our work we have focused on the development of the behavior-definition mechanism (and corresponding format) which:

- will be suitable for the simulation of the power distribution network features and problems
- will profit of the data gathered in the traditional power network failure databases

Below we will briefly present the main outlines of our system.

3 THE SIMULATOR

Our simulator, whose prototype is called Rice [9], is based on the simulation of real power distribution network facilities by individual agents. Topology of the network is based on the ad hoc established inter-agent information flow. With such a design we can simulate network with dynamic topology and even dynamically add and remove its components or change their behavior.

3.1 Technical Overview

Rice is built using the distributed approach with agents to computers in M:N possible combinations which allows us to use different deployment strategies varying from the whole system installed on one computer up to the highly-distributed scenario with one resource-consuming agent on one dedicated machine.

The communication between agents is based on the CORBA/TCP location [10] and remote procedure invocation layer combined with the standard KQML/content language semantic layer [11]. This environment is platform and language independent, making it possible to have different parts of the system implemented in different programming languages and even on different operating systems. For our daily development we use Linux and MacOS with Java and Python.

At present time we are examining the usage of the self-contained system image based on the Linux operating system which can be used as a virtual machine image or a cluster-node image for the deployment of the system in the highly distributed scenario, see the Figure 1.

3.2 Agents

The simulator is based on individual entities called agents. Every agent is a standalone program which uses predefined library primitives for constituting its behavior. The behavior definition is designed as a collection of simple actions triggered by the changes of the agent knowledge base and changes of its surrounding environment. From this point of view the current state of the simulator is based on so-called *reactive agents*, but future development into more complex agents is open.

The main behavior components are:

- *Subscription* – mechanism used for subscribing for some knowledge of another agent. When the knowl-

```
<agent class="BasicFacility" type="A_RL_NODE" name="N1">
<init>
<knowledge name="lastPowerFlow" value="0"/>
<knowledge name="powerFlow" value="0"/>
<knowledge name="maxPowerFlow" value="10000"/>
</init>
</agent>
```

Fig. 2: The XML definition of the properties of the source node.

edge changes, the subscriber is informed. E.g. agent B subscribes for the knowledge about the status of power output of agent A. This means that agent B will be automatically informed about every change of characteristics of the output voltage of the agent A. When outage comes to the agent A, agent B can decide to propagate the outage or to switch to another input. Unsubscription of the knowledge means that agent is no more interested in its value and its propagation is stopped. Such an action can be done e.g. if the network topology is to be changed and agent A is no more the input of the agent B.

- *Message handlers* – these functions are triggered when some defined messages arrive at the agent.
- *In-reply-to handlers* – used for filtering of the arriving messages, preferably for reacting to answers to the previously sent questions and subscriptions
- *Missing value generators* – function used for generating some values which are not present in the agent knowledge base
- *Knowledge generators* – function generating some particular knowledge value. It is triggered every time the knowledge is accessed.
- *Knowledge change handlers* – function triggered whenever the value of its associated knowledge changes.
- *Timer event* – single or recurring time-based events
- *Event database* – the handlers can use built-in event database methods to save knowledge values for off-line data postprocessing (statistical analysis, chart generations, etc.) The gathered data can be exported to standard formats.

These behavior definition components are designed to be independent pieces of code and thus the same functions can be assigned to the different agents. The overall behavior of the particular agents (and therefore the behavior of the whole simulator) is constituted from the configuration and communication of these small primitives.

3.3 XML Format and Machine Generated Behavior

The principle of the independent separated behavior items design was led by the motivation to develop a behavior definition for agents which could be computer-generated on the basis of the real-life gathered statistical data from the already existing failure databases. For making this cooperation possible we have developed an XML format for describing the behavior elements set ascribed to the particular agents in the network.

At present this format is used for defining the above mentioned handlers (by embedding Python code directly in the XML) and defining which elements are to be used in which agents. The format also specifies the initial state of the knowledge bases of the agents, building up the input for the action/reaction handler codes. Thus the parameters of the agents behavior are based on the statistical data gathered in the failure databases.

3.4 Example

In our example we will illustrate that the implementation of a specific problem in the Rice simulator is really the matter of a *behavioral definition of the agents*, not classical programming. We will show just a few handler code examples. For more complex and detailed example please see [12].

To build the network of agents simulating power flow, outages and non-delivered energy cost counting, we must define 4 types of the agents:

- power source
- power line
- network node (distribution point)
- power consumer

There will be a flow of energy demand messages from the consumer nodes to the source and energy granted messages vice versa. Outages will be propagated automatically by the subscription mechanism. The outage cost will be counted on the consumer nodes.

To accomplish this we will define these actions and properties:

- The *power source* agent will have only one `init` property: maximal deliverable energy amount. If this will be exceeded, the agent will fall into *outage* state and no energy will be delivered further.
- The *power line* will just copy its input energy characteristics to its output and copy demands from its output to its input. Subscription mechanism will be used.
- The *distribution point* will sum the demands from its outputs and propagate the sum value to its input.
- All agents must propagate the outage event.
- To simulate the non-delivered energy cost, every agent will have its loses function which will be triggered on the outage begin and in its end it will return overall outage cost.

The XML definition of the properties of the source node are presented in the Figure 2. The handler for the changes of the power flow is defined in the Figure 3.

Note that no actions which should be taken according to the outage ("`rlStatus=error`") are defined here –

```
<knowledgeBase>
    <handler valueName="powerFlow">
        <action type="python">
            def action(self,data,*args,**kwargs):
              if self['powerFlow'] &gt; self['maxPowerFlow']:
                  self['rlStatus']='error'
        </action>
    </handler>
</knowledgeBase>
```

Fig. 3: The definition of the handler that checks the amount of delivered energy of the source node (note that ">" here means ">" in XML).

```
<action type="kqmlMessage">
    <kqml performative="subscribe" language="Kqml"
      receiver="$self['inputName']$">
        <kqml performative="ask" language="Trivial"
        receiver="$self['inputName']$" reply-with="input status">
            rlStatus=?
            rootCause=?
        </kqml>
    </kqml>
</action>
```

Fig. 4: An example of a node's subscription message to receive information from another node.

they are defined in the other handler and are triggered automatically when the value of the "`rlStatus`" knowledge changes.

Also informing the agent which is connected to the output of this agent is not defined here – it will be automatically informed about the status change because he is a subscriber of the "`rlStatus`" value. An example of its subscription message is presented in the Figure 4. This definition means that the agent wants to be informed about every change of the values of `rlStatus` (condition of the node) and `rootCause` (root cause of the failure) in the knowledge base of the agent which is to be its input. Note that only changing of the message recipient will be sufficient to change the input agent and thus the network topology. We can change the topology dynamically by just changing the recipients of the subscriptions.

Whenever the status of its input changes, the agent will be informed about that and can react to this information with an appropriate handler. Handler is triggered by all messages with "input status" value in the `in-reply-to` field – see this value in the above subscription definition. This handler is defined in the Figure 5. You can see that again the handler only changes the values of `rlStatus` and `powerFlow`. This will trigger other subsequent actions.

4 INTERACTING WITH THE FAILURE DATABASES AND OTHER REAL-LIFE DATA

To make the simulation process more valuable, we can use real-life gathered data. At present time we focus our research on the matter of using data from the power distribution network failure databases as an input for our simulations. The main guidelines of it will be briefly outlined.

4.1 Failure Statistics

The failure databases gather the data about the failures taking place in the power distribution network. Rice simulator enables us to use the data about *failure rate* of the particular node type as an input for the time-based internal event handlers (failure generators). These generators also use the values of the *failure duration* and its *type*. Tracking of the consequences and propagation of the failures is done by the mechanisms described formerly.

4.2 Power Consumption and Its Development

Each consumer node has a *consumption function* which is used in the power demand generator. The (changing) power demand is propagated through the network and it is compared with the distribution abilities of the particular nodes, causing failures or not. The actual power demand placed on the particular node is measured in real-simulated-time and not preliminary defined, thus it can be a complicated function of the (possibly dynamically changing) network topology.

4.3 Non-delivered Energy Cost

Based on the real-life observation, we can define a *loss function* of particular nodes, which is used for counting the overall losses of the consumer nodes based on their type. Again the standard Rice knowledge change handling mechanism is used for triggering of the loss function for a particular agent. Storing loss values to the event databases allows the system to make a summary and graphical presentation of all losses at the end of the simulation.

This way we can get summary loss for the network even if it dynamically changes its parameters during the simulation process, which could be hard to get if we had used a computation-only approach.

```
<handler inReplyToFilter="input status" theOnlyAction="yes">
    <action type="python">
        def action(self,data,*args,**kwargs):
            messageContent=data['messageContent']
            inputStatus=messageContent['rlStatus']
            if inputStatus =='outage' or inputStatus =='error':
                self['rlStatus']='outage'
                self['powerFlow']=0
    </action>
</handler>
```

Fig. 5: An outage handler definition.

5 CONCLUSIONS

We have been carrying out the observing and analyzing of failures from the majority of distribution areas from the Czech Republic and one from the Slovak Republic at the research workplace of the Technical University of Ostrava since the year 2000. We have tried to unify data processed to enable the comparison of results and to create a statistically more important database.

The result of analyses does not have to be only the determination of the rates and the mean durations of failures for particular items of equipment, but also the obtaining of other pieces of information being important for operators, such as the oftenest cause of failures, areas of the greatest amounts of unsupplied energy, etc. The larger the data range, the more accurate will be statistical results. That is why we would be glad to have also a possibility to enhance the databases by data from other countries.

ACKNOWLEDGMENTS

The research has been partly supported by the Academy of Sciences of the Czech Republic – project No: AVČR 1ET100300414.

REFERENCES

[1] R. E. BARLOW AND F. PROSCHAN: *Statistical Theory of Reliability and Life Testing: Probability Models*, Holt, Rinehart and Winston, Inc., 1975.

[2] N. R. MANN, R. E. SCHAFER AND N. D. SINGPURWALLA: *Methods for Statistical Analysis of Reliability and Life Data*, John Wiley and Sons, 1977.

[3] J. ENDRENYI: *Reliability in Electric Power Systems*, John Wiley and Sons, 1978.

[4] W. H. BEYER: *Standard Mathematical Tables*, 26th Edition, CRC Press, Inc., 1981.

[5] R. RAMAKUMAR: *Engineering Reliability: Fundamentals and Applications*, Prentice-Hall, Inc., 1993.

[6] T. GÖNEN: *Electric Power Distribution System Engineering*, McGraw Hill, 1986.

[7] S. ASGARPOOR AND M. J. MATHINE: Reliability Evaluation of Distribution Systems with Non-Exponential Down Times, In *IEEE/PES 1996 Winter Meeting*, Baltimore, MD, IEEE, 1996.

[8] R. E. BROWN, J. R. OCHOA: Distribution System Reliability: Default Data and Model Validation, In *IEEE Transaction on Power Systems*, Vol. 13, No. 2, May 1998, pp. 704-709.

[9] MIROSLAV PRÝMEK AND ALEŠ HORÁK: New Features in Power Networks Modelling Using the Rice System, In *Proceedings of ElNet 2006* , Ostrava, 2006. VSB Technical University of Ostrava.

[10] OBJECT MANAGEMENT GROUP: *Common Object Request Broker Architecture Core Specification.*, 2004, `http://www.omg.org/technology/documents/formal/corba_iiop.htm`.

[11] T. FININ, R. FRITZSON, D. MCKAY, AND R. MCENTIRE: KQML as an Agent Communication Language, In *Proc. of* 3$^{\text{rd}}$ *Int. Conf. On Information and Knowledge Management CIKM 1999*. ACM Press, 1994.

[12] PRÝMEK, M., HORÁK, A.: XML as an Agent-Behaviour Definition Language, In *Proceedings of ElNet 2007*, pp. 57–67, Ostrava : VŠB – Technical University of Ostrava, 2007.