SMARTRESOURCE PLATFORM IN DISTRIBUTED POWER NETWORKS MAINTENANCE

DELIVERABLE 3.2

University of Jyväskylä, Agora Center
October 2006

Author: Industrial Ontologies Group

Contact Information*:* e-mail: vagan@it.jyu.fi

Title: SmartResource platform in distributed power networks maintenance

Work: Technical report

Number of Pages: 28

Abbreviations

RFID – Radio Frequency Identification

IT – Information Technology

GUN – Global Understanding eNvironment

DMS - Distribution Management System

RAB - Reusable Atomic Behavior

RGBDF – Resource Goal/Behavior Description Framework

JADE - Java Agent Development Environment

GML - Geography Markup Language

KML – Keyhole Markup Language

XML - eXtensible Markup Language

HTML – HyperText Markup Language

LAN - Local Area Network

WAN - Wide Area Network

RDF – Resource Description Framework

API - Application Programming Interface

Contents

1 Introduction

Recent advances in networking, sensor and RFID technologies allow connecting various physical world objects to the IT infrastructure, which could, ultimately, enable realization of the "Internet of Things" and the ubiquitous computing visions. This opens also new horizons for industrial automation, i.e. automated monitoring, control, maintenance planning, etc, of industrial resources and processes. A much larger, than in present, number of resources (machines, infrastructure elements, materials, products) can get connected to the IT systems, thus be automatically monitored and potentially controlled. Such development will also necessarily create demand for a much wider integration with various external resources, such as data storages, information services, and algorithms, which can be found in other units of the same organization, in other organizations, or on the Internet.

Such interconnectivity of computing and physical systems could, however, become the "nightmare of ubiquitous computing" in which human operators will be unable to *manage* the complexity of interactions, neither even architects will be able to *anticipate* and *design* that complexity. It is widely acknowledged that as the networks, systems and services of modern IT and communication infrastructures become increasingly complex, traditional solutions to manage and control them seem to have reached their limits. The IBM's vision of autonomic computing proclaims the need for computing systems capable of "running themselves" with minimal human management which would be mainly limited to definition of some higher-level policies rather than direct administration. The computing systems will therefore be *self-managed*, which, according to the IBM vision, includes self-configuration, self-optimization, self-protection, and self-healing. We believe that such self-manageability of a complex system requires its components to be to a certain degree autonomous themselves. In other words, we envision that agent technologies will play an important part in building such complex systems. Agent-based approach to software engineering is also considered to be facilitating the *design* of complex systems.

Another problem is inherent *heterogeneity* in ubiquitous computing systems, with respect to the nature of components, standards, data formats, protocols, etc, which creates significant obstacles for interoperability among the components of such systems. Semantic Web technologies are viewed today as a key technology to resolve the problems of interoperability and integration within heterogeneous world of ubiquitously interconnected objects and systems. The Internet of Things should become in fact the *Semantic Web of Things*. Our work subscribes to this view. Moreover, we believe that Semantic Web technologies can facilitate not only the discovery of heterogeneous components and data integration, but also the behavioral coordination of those components.

In the SmartResource project (2004-2006), we worked on the *Global Understanding Environment (GUN)*. GUN is a general middleware framework aiming at providing means for building complex industrial automation systems consisting of components of *different* nature, based on the Semantic Web and agent technologies. The word "global" in GUN has a double meaning. First, it implies that industrial resources are able to communicate and cooperate globally, i.e. across the whole organization and beyond. Second, it implies a "global understanding". This means that a resource A can understand all of (1) the properties and the state of a resource B, (2) the potential and actual behaviors of B, and (3) the business processes in which A and B, and maybe other resources, are jointly involved.

This report describes a case study in the domain of distributed power network maintenance we have been performing, starting from early 2006, in collaboration with ABB Company (Distribution Automation unit). The goal is to study the potential add-value which ABB could receive from introducing Semantic Web technologies and Global Understanding

Environment (GUN) framework in particular into their business. Development of a prototype, for demonstration of the concept purposes, was a part of the study as well.

Section 2 provides a very brief description of the domain and follows with a vision of potential new functionality and applications that could be created based on GUN. Section 3 presents the presents state of the SmartResource platform, which is a pilot implementation of GUN. Section 4 reports then on the prototype developed for the power networks domain, which also demonstrates some of the basic features of the SmartResource platform itself.

2 The Vision

A very brief description of the domain follows. A basic unit of monitoring in a power network is a *feeder*, which is a section of the power line including all the poles, conductors, insulators, etc. The start and the end point of a feeder are *substations*, whose task is to transform the electric power e.g. from high-voltage to medium-voltage or from medium-voltage to low-voltage. In addition to the transformer, any substation naturally includes the devices monitoring and protecting both the incoming and the outgoing feeders. Such *protection relays* automatically monitor the state of the feeder in terms of voltages and currents, are able to disconnect the feeder if a significant *disturbance* is registered, and to automatically re-close the circuit after a specified time (and to break it again if the disturbance persists).

Persistent disturbance is usually a sign of a *fault* in the network, which could be e.g. earth fault (conductor falling of the ground), short-circuit (could be caused e.g. by a tree falling on a line with bare conductors), or open circuit (broken line). Restoration of the network, after a fault occurs, includes *fault detection*, *fault localization* (estimating the geographic location of the fault), and of course fault removal. In meanwhile, network reconfiguration may also be performed, with a goal of e.g. minimizing the number of customers who will suffer outage of power until the fault is removed.

As mentioned, the fault detection is performed by protection relays. The rest is performed in the *operation centers* with participation of human *operators*. In case of a fault, protection relay sends an alarm to the operation center and also sends a dataset with recorded disturbance: several-second history of all the monitored parameters with a high frequency of sampling (0.5 ms or so). A certain operation center controls a sub-network of the integral power network. The operators use systems like ABB Distribution Management System (DMS) to have an integrated graphical view over the sub-network and ABB MicroSCADA for data acquisition from the substations and remote control over the relays, switches, etc. The systems like DMS also include implementations of various algorithms: for fault localization, for calculation of optimal reconfiguration of the network and other.

ABB is a vendor of hardware and software for power networks. The medium-voltage sub-networks are owned, controlled and maintained then by some local companies, e.g. Jyväskylän Energia for the city of Jyväskylä, and Vattenfall for all the rural areas around. It is noticeable that the operation centers of different companies have no connection to each other, so information exchange among them is nearly impossible. In the case of a fault affecting two different sub-networks, such information exchange, though, may be very important, for all of fault localization, network reconfiguration, and network restoration. Introducing an inter-organizational SmartResource platform could solve this issue (Figure 1). The information flow will go through the agents representing the sub-networks on the SmartResource platform. Utilization of Semantic Web technologies will allow such interoperability even if the sub-networks use software systems from different vendors (ABB is not the only one), and thus maybe different data formats and protocols.

The second scenario in our vision is related to *a new business model* that ABB could implement. At present, all ABB expertise gets embedded into hardware or software systems and sold to the customers as it is. A new business model would be to start own Web-service providing implementation of certain algorithms, so the ABB customers will utilize those algorithms online when needed (Figure 2). ABB will be always able to update algorithms, add new, and so on.

Fig. 1. Scenario: sub-networks interoperability

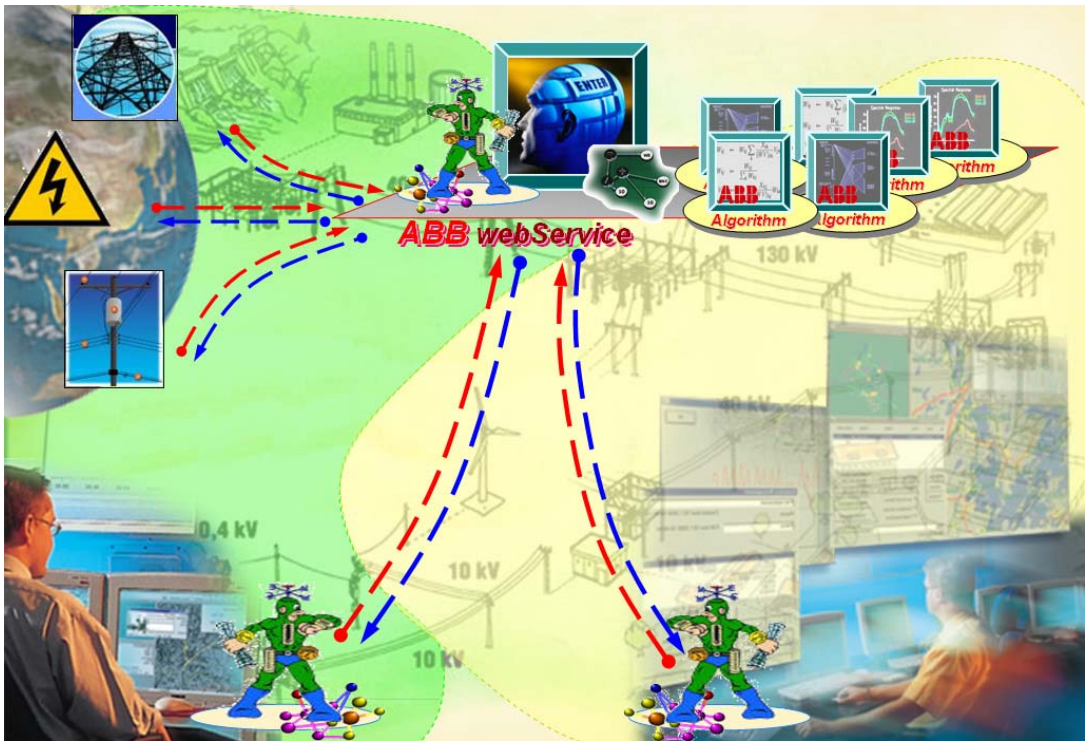

Fig. 2. Scenario: a new business model

SmartResource platform will ensure interoperability and coordination between such Web-service and customers' software systems, and also a relative ease of implementation of such a solution – because it will not require changes in existing software systems, only extension with the SmartResource platform. Noticeable that, if semantically defined, such Web-service

can potentially be utilized across the globe even by the customers who never purchased any of ABB hardware or software.

Fig. 3. Scenario: integration with external information services

The third scenario in our vision is related to the possibility integrating data, which is currently utilized in the power network management (network structure and configuration, feeder relay readings), with contextual information from the external sources (Figure 3). Such integration can be used for:

- *Risk analysis*. Information about whether conditions, ongoing forest works, or forest fires can be used for evaluating existing threats for the power network. This may be used to trigger an alert state for the maintenance team, or even to do a precautionary reconfiguration of the network to minimize possible damage.
- *Facilitation of fault localization*. The output of fault localization algorithms is not always certain. The information about threats for the power network that existed at the time when the fault occurred (which thus may have caused the fault) may greatly facilitate the localization. In some situations, contextual information alone may even be sufficient for localization.
- *Operator interface enhancement.* Contextual information may be used also just to extend the operators' view of the power network. For example, satellite imagery can be used for geographic view (instead of locally stored bitmaps as it is in the current DMS); also, dynamically-changing information can be accessed and represented on the interface.

The last scenario is our vision is about the possibility of transferring the knowledge of human experts to automated systems, by means of various data mining tools (Figure 4). In the power network management case, one scenario that seems to be highly appropriate for such knowledge transfer is the following. In present, it is always a decision of a human expert which of the existing fault localization algorithms will perform the best in the context of the

current configuration of the power network and the nature of the fault. Such decisions made by an expert along with the input data could, be forwarded to a learning Web-service. After a sufficient learning sample, this Web-service could start to be used in some situations instead of the human expert, e.g. in situations when a faster decision is needed or when the expert is unavailable.

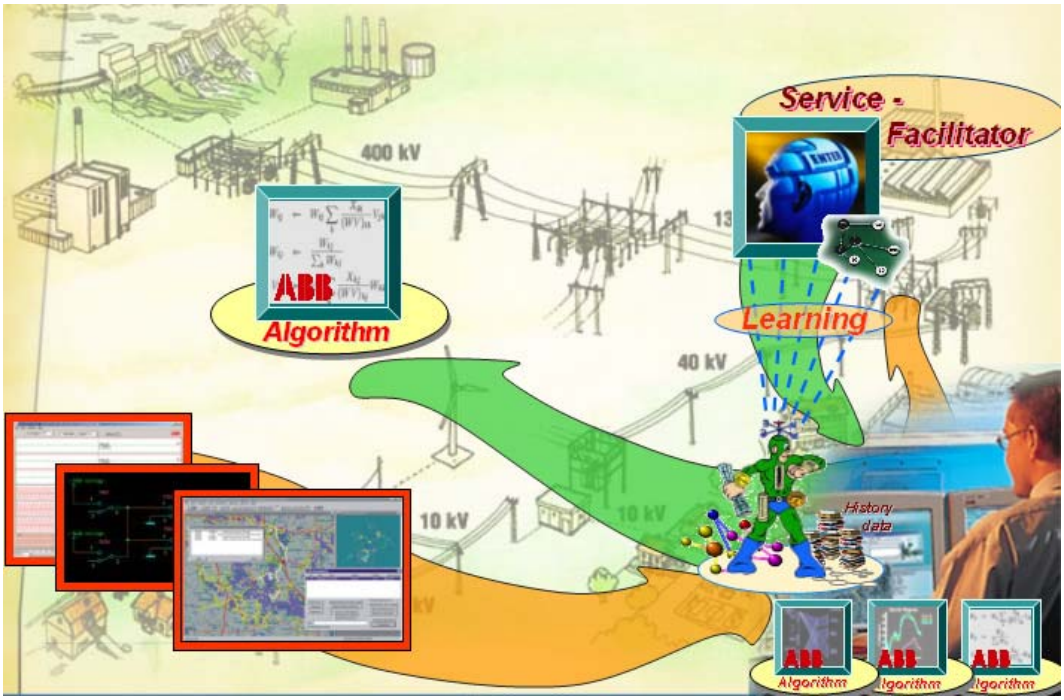

Fig. 4. Scenario: expert's knowledge transfer

3 The SmartResource Platform

This section describes the current state of the SmartResource Platform. The central to the platform is the architecture of a SmartResource agent depicted in Figure 5. It can be seen as consisting of three layers: reusable atomic behaviors (RAB), behavior models corresponding to different roles the agent plays, and the behavior engine.

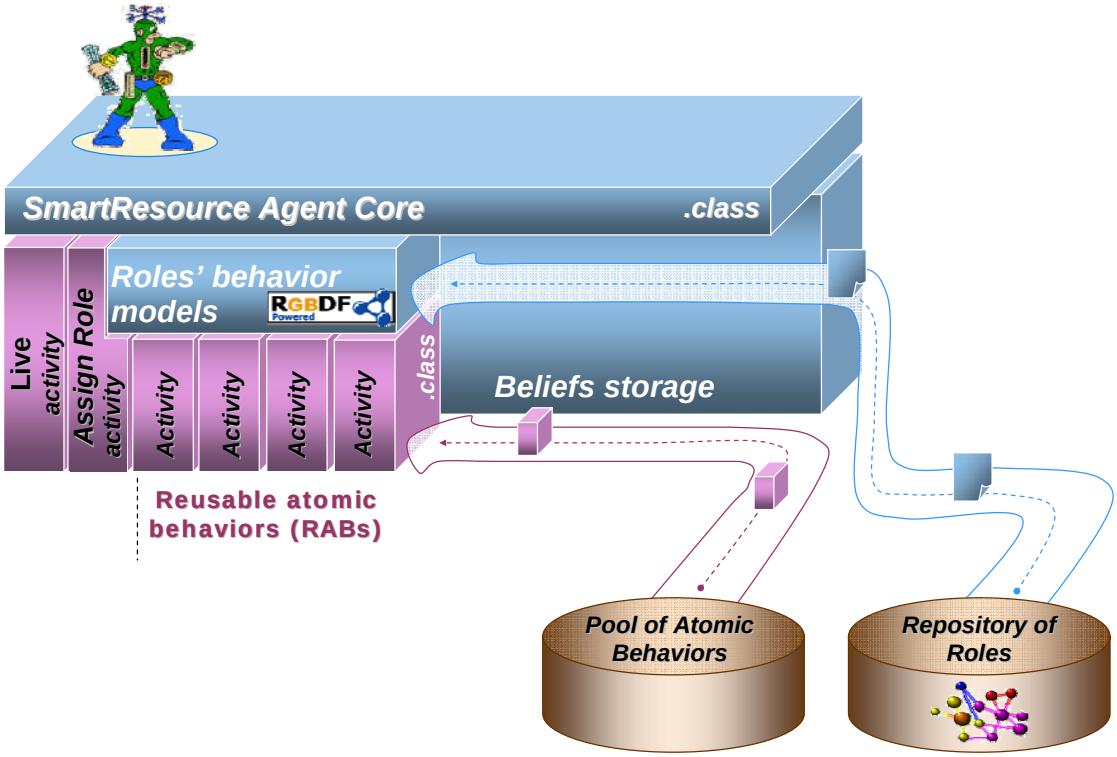

Fig. 5. SmartResource agent architecture

A *reusable atomic behavior (RAB)* is a piece of Java code implementing a reasonably atomic function. Therefore, RABs correspond to actuators and preceptors of the agent. As the name implies, RABs are assumed to be reusable across different applications, different agents, different roles and different interaction scenarios. Some examples of RABs from our case system (see the next section) are:

- RequestSenderBehavior and RequestReceiverBehavior – actuator and preceptor, correspondingly, related to sending and receiving requests for some service.
- DataSenderBehavior and DataReceiverBehavior – actuator and preceptor, correspondingly, related to sending and receiving the responses upon requests.
- ExternalApplicationStarterBehavior – popping-up an external application in order, e.g., to visualize some information to a human.
- DirectoryLookupBehavior – interacting with a system agent called directory facilitator (DF), which stores the mapping between agents and roles, in order to find agents playing a specific role.
- RandomSelectBehavior – randomly selecting an agent among several playing the same role.

- OneStepAuctionBuyerBehavior and OneStepAuctionSellerBehavior –implementing the buyer and the seller behaviors of the simplest auction: request for bids is sent, bids from all are received or time limit is expired, and the best bid is selected.

In the SmartResource Platform, the behavior of an agent is defined by the roles it plays in one or several organizations. Some examples of the possible roles: operator's agent, feeder agent, agent of the feeder N3056, fault localization service agent, ABB fault localization service agent, etc. Obviously, a general role can be played by several agents. On the other hand, one agent can (and usually does) play several roles.

A *behavior model* is an RgbDF document that is supposed to specify a certain organizational role, and, therefore, there is one-to-one relation between roles and behavior models. A behavior model consists of a set of beliefs representing the knowledge needed for playing the role and a set of behavior rules. Roughly speaking, a behavior rule specifies conditions of (and parameters for) execution of various RABs. Obviously, RABs need to be parameterizable. For example, RequestSenderBehavior takes such parameters as the agent to send the request to and the request itself. Notice that, in SmartResource Platform, if a behavior model specifies the need of interaction with another agent, that agent is always specified by its role, not name or another unique identifier of a particular agent. If several agents play the role needed, the behavior model is supposed to include some rules specifying a mechanism of resolving such a situation, e.g. random select, auction, etc. Different such mechanisms can of course be assigned to resolving conflicts with respect to different roles.

The *behavior engine* is the same for all the SmartResource agents (we mean that each agent has a copy of it). The behavior engine consists of the agent core and the two core activities that we named "assign role" and "live". The AssignRole activity is responsible for parsing RgbDF of a behavior model into the beliefs and behavior rules storages. The Live activity implements the run-time loop of an agent. Roughly speaking, it iterates through all the behavior rules, checks them against current beliefs and goals, and executes RABs corresponding to the rules to be fired. Upon creation of the agent, the AssignRole activity needs to be invoked directly from the agent's core to parse the startup behavior model; however, all the later invocations of AssignRole (parsing of actual roles) are made according to that startup model. Therefore, AssignRole has the duality of being a part of the engine and a RAB in the same time.

As can be seen from Figure 5, in SmartResource Platform, agents access the behavior models from an external repository, which is assumed to be managed by the organization which "hires" the agents to enact those roles. It is done either upon startup of an agent, or if the organization requests an update to be made. Such externalization of behavior models has several advantages:

- Increased flexibility for control and coordination. Namely, the organization can remotely affect the behavior of the agents through modifying the behavior models. Another advantage is that the models can always be kept up-to-date.
- An agent may 'learn' how to play a new role in run-time; it does not need to be pre-programmed to do it.
- Inter-agent behavior awareness. How is discussed in the previous section, the agents not enacting a particular role can still make some use if the information encoded in its behavior model. One reason is to understand how to interact with, or what to expect from, an agent playing that role.

As can also be seen from Figure 5, SmartResource Platform allows on-demand access even of RABs. If an agent plays a role, and that role prescribes it to execute an atomic behavior that the agent is missing, the agent can download it from the repository of the organization.

In a sense, the organization is able to provide not only instructions what to do, but also the tools enabling doing that. The obvious additional advantages are:

- An agent may 'learn' new behaviors and so enact in a completely new role.
- Agents may have a "light start" with on-demand extension of functionality.

Technically, SmartResource Platform is implemented on the top of the Java Agent Development Environment (JADE). Therefore, the SmartResourceAgent is a subclass of jade.core.Agent. All the RABs have to be subclasses of some of the subclasses of jade.core.behaviours.Behaviour, e.g. OneShotBehaviour, CyclicBehaviour, and so on.

4 Current ABB Prototype

We also developed a prototype, mainly for the purpose of the concept demonstration, both for ABB and their customers. The current prototype includes the following smart resources, represented by the corresponding agents:

- *Operator*. A human operator monitoring and controlling the power network. In addition to the traditional interfaces – DMS/MicroSCADA – the operator is provided with an additional interface by the operator' agent (see below).
- *Feeders*. Each feeder (section of the power network) is represented by an agent. Those agents are responsible for answering operator's requests for the current state of the feeder, and also for sending alerts when a disturbance is registered. Technically, feeder agents are accessing feeders' data from the MicroSCADA system.
- *Network Structure Storage*. The DMS system is utilizing a database for storing the data on the power network including the network graph structure, detailed data on substations, feeders, etc. The network storage agent is responsible for interaction with that database for answering operator's requests for the network graph (for visualization) and e.g. for detailed data on a substation.
- *Fault Localization Services*. We assume the scenario presented in Figure 2 will be eventually realized. The fault localization can be then performed by an external entity, e.g. a Web-service, which will also be represented on GUN platform by a corresponding agent (the service itself is a stub in the prototype).
- *Weather Service*. A service providing current weather conditions and forecast for a geographic location. We utilized one provided by the Finnish Meteorological Institute.
- *Forest Fire Alert Service*. A service that is supposed to issue alerts when there is a forest fire (a stub in the prototype). The agent representing this service is responsible for automatic forwarding such alerts to the operator's agent.
- *Geographic Service*. Provides the geographic map data in Geography Markup Language (GML), if operator's agent requests.
- *Repository of Roles* and *Pool of Atomic Behaviors*. See Section 3.

In the current prototype, both the repository of roles and the pool of atomic behaviors are managed by the same agent with the role "OntologyAgent". Also, there is only one single repository of the roles, which is also, in fact, a simplification. Consider the scenario of the fault localization by an external service. The agent representing such a service has to necessarily play at least two different roles. One is "our localization service seller" for the company developed the service, say, ABB. The other is "localization service agent" for the company running a power network, say, Jyväskylän Energia. It is because the agent needs to represent the interest of ABB, sell the service for them; but it is also obliged to deliver the service according to the rules, protocol, etc. specified by Jyväskylän Energia. Obviously, it is reasonable that each of cooperating organizations will maintain its own repository of the roles it defines. However, for a prototype, implementing this was not so important.

Figure 6 shows the process of starting up an agent. The same process is followed for every new agent on the GUN platform. From the startup batch file of the GUN platform, an agent receives only the names of the roles that it has to play (the same holds also for cases when an agent in created in run time). For the example in Figure 6, the agent called "feeder1" gets to know that it has play the general role "FeederAgent" – common for all the feeder agents, and a particular role "FeederID1" – needed for that other agents will associate this agent with the feeder ID1, and including a set of beliefs and rules specific for getting connected to and managing that particular feeder. First, the agent "feeder1" loads the

startup.rdf script, which is again common for all the agents. According to that script, the agent contacts the Directory Facilitator to find the agent who plays the "OntologyAgent" role. The Directory Facilitator maintains a mapping between agents and roles they play. After the OntologyAgent named "ontology" is discovered, it is contacted and asked to deliver the two scripts, one per role needed. After the delivery, "feeder1" loads the scripts and starts to work according to them. It also registers itself with the Directory Facilitator, so that other agents will be aware that it now plays those roles.

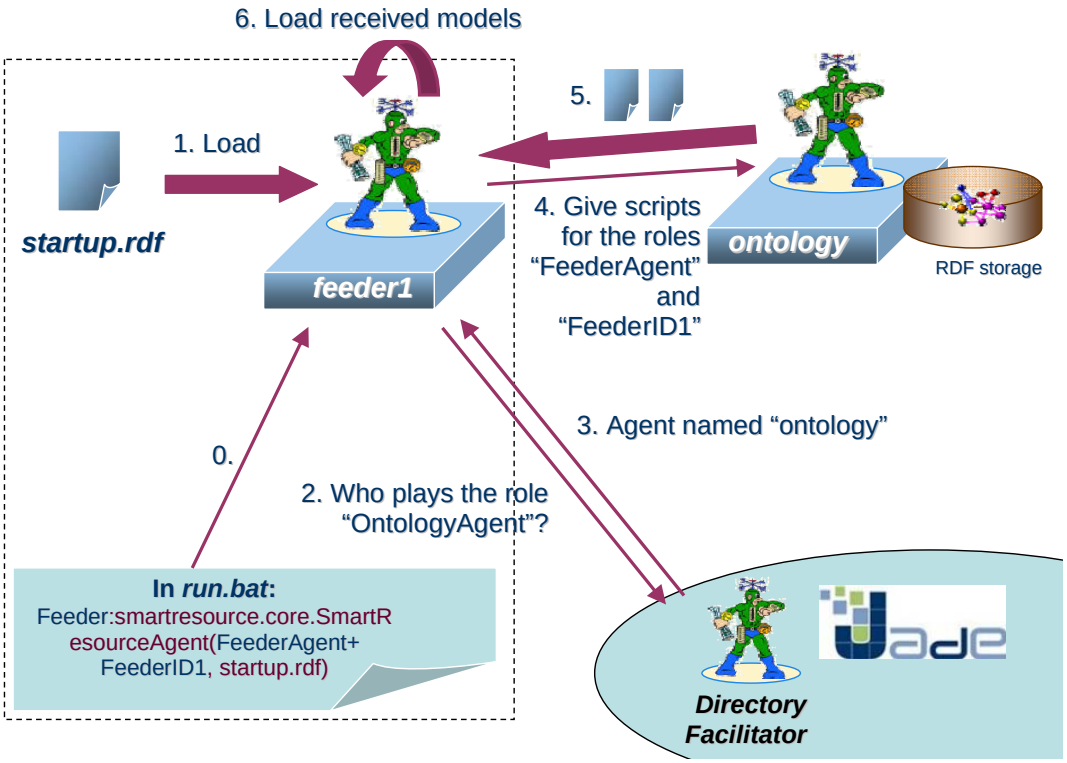

Fig. 6. An agent's start-up

Figure 7 depicts a more complex scenario of auction for selection of a service provider, in this case a fault localization service. Using the Directory Facilitator, the operator's agent discovers that there are two competing agents on the platform that provide the fault localization service. The operator's agent checks its script for a rule resolving such a situation and discovers that, in case of several localization services, an auction has to be performed (for other roles, random select is done). The agent first sends to both localization agents a special request "Load Role OneStepAuctionSeller", and then a request to make an offer on, say, price of the service. The agent "ls1" has loaded the role "OneStepAuctionSeller" from the beginning, but the agent "ls2" did not. So, "ls2" contacts the OntologyAgent and requests the needed script now. A simple check of rights is performed just before that: with the Directory Facilitator "ls2" checks whether the requesting agent "operator" is working in the role that empowers it to make this particular request, "OperatorAgent" in this case. The agent "ls1" makes its offer immediately, while "ls2" does that after it gets the script and, likely, the corresponding RAB. Then, the operator's agent selects one of the providers and commits the service transaction with it. This scenario demonstrates that roles can be loaded also dynamically.

Obviously, "ls1" and "ls2" needed to enact the "LocalizationService" role earlier. The behavior model corresponding to it will enable the agent to actually deliver the service in the step 12. Also, "ls1" and "ls2" needed to enact some roles like "our service seller" of the corresponding service provider organization. The behavior models of those roles are the places from which they, e.g., get such information as what price to ask from the clients.

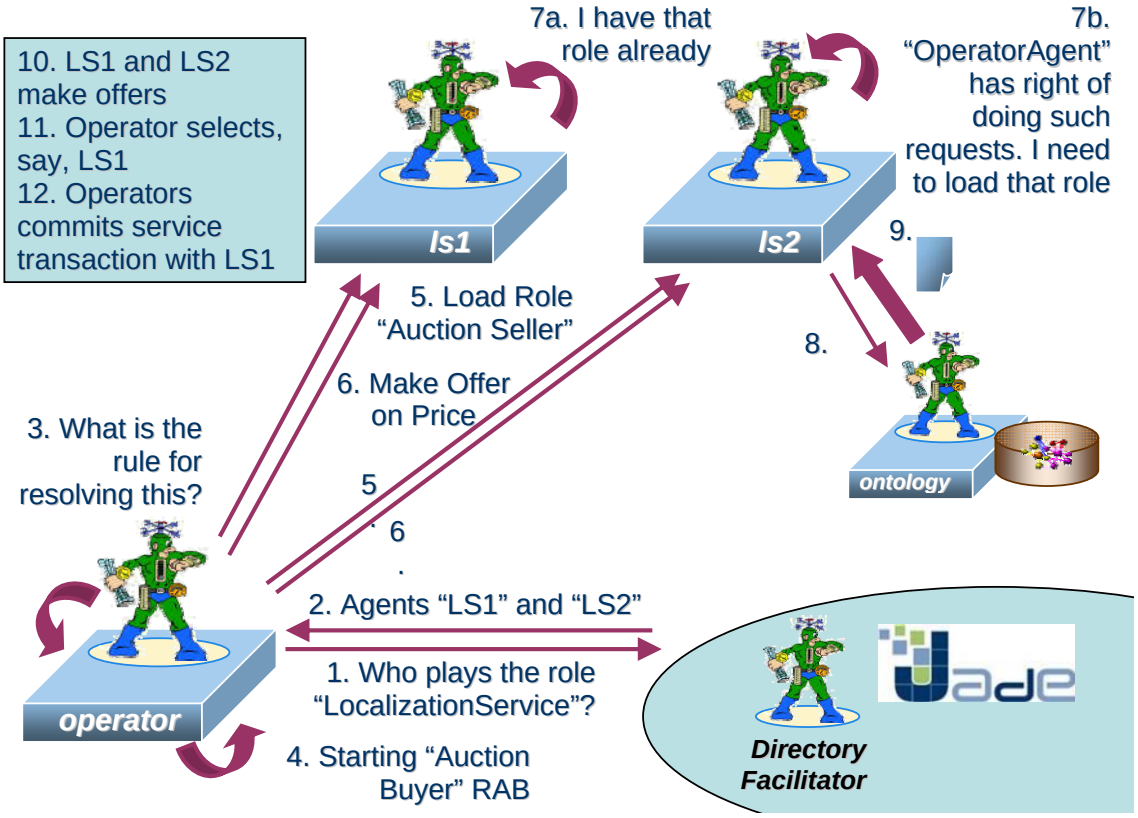

Fig. 7. Auction for selection of the service provider

Figure 8 shows the interface of an operator generated by the operator's agent. The interface consists of the following elements. First, there is a small command window with buttons "Show network in GML", "Show network in GoogleEarth", "Request localization service" and "Send maintenance crew". Second, there is the main graphic interface, which comes in two options. One option utilizes a freeware GML viewer. The other option utilizes the GoogleEarth application, which uses Google's own KML language for defining data to be overlaid over the map. Both GML and KML are XML-based markups, so transition is easy. In the case of using GoogleEarth, participation of the Geographic Service agent is, obviously, not required. The advantage of using GML map data, though, is that it can be used as input for some analysis if needed. For example, one could wish to estimate how the forest fire can progress with time – the information about where lay the boundaries of forests and open spaces or lakes is then important, and may be encoded in GML. In contrast, a satellite image will provide little help in that case.

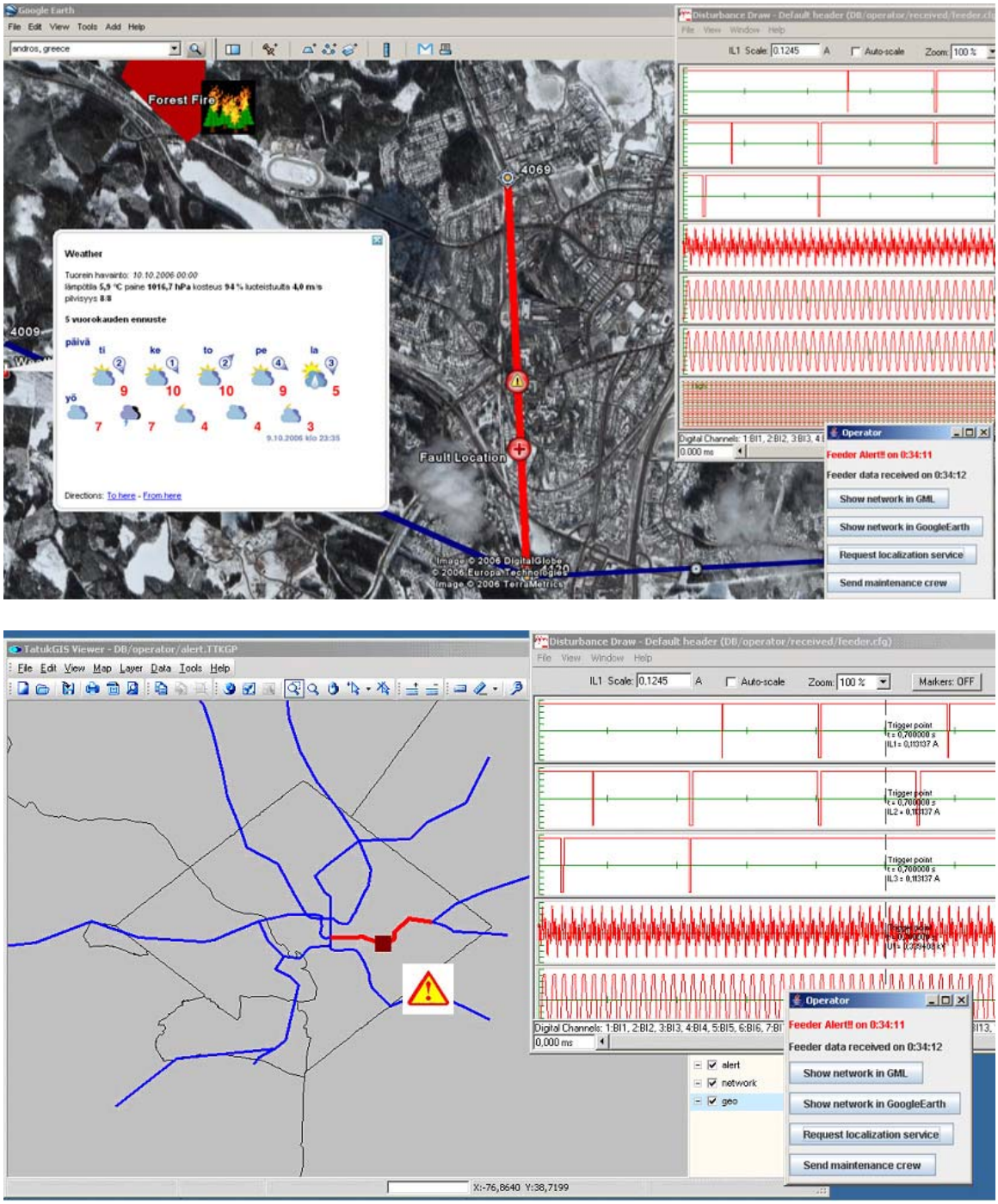

Fig. 8. Interface of an operator provided by his/her agent (2 versions)

Finally, the interface may include some other external applications that the operator's agent can pop-up when needed. So, using the main graphic interface, the operator can request the data on the current state of a feeder, and the data delivered by the corresponding feeder agent is visualized using the ABB Disturbance Draw application. The operator can also request detailed description of a substation, which will be represented with HTML in an Internet browser window.

5 Details on integrating with existing ABB products

5.1 Background

By integrating the typical functions of power network into operator dashboard, such as fault diagnosis, control, monitoring, on-line restoration, maintenance [1], the accurate and centralized approach to the operation or can be guaranteed without switching between several applications. To access the same information today, requires that several applications from different products are accessed. In the case of fault occurring in the power network, and there is need to analyze the disturbance recordings from the protection and control units, a dedicated application is manually started by the operator. To access the monitoring and controlling application, the graphics of the appropriate single-line diagram is presented in the separate MicroSCADA system. If additional simulation functions are needed by the operator, the Distribution Management system is launched.

As the text above describes, the operator requires launching or interacting with several applications to perform, which is seen as de-centralized today when performing operations. From the operator point of view, different criterion for the decision making may apply. One may be to minimize the power shortage time for the consumers. This may become guaranteed also that for the most critical switches of power network, there may be even automatic restoration operations performed by the power distribution systems. Another decision criterion for the operator may be to maximize the amount of consumers, which become re-connected to the power network. Additionally, one decision criteria may be the fact that the power shortage may result severe consequences to the safety related organizations (hospitals), or the raw material in the production process of assembly line results into non-current state (metal industry companies).

As the text above describes, there is evidently requirements to get the relevant information out of the underlying devices and systems into centralized dashboard to assist the operator actions in these circumstances. Next, these enablers for expected operator dashboard are briefly described including their role in the power distribution process.

Protection & control devices interact with the equipment of the power network; such as circuit breakers, disconnections, earth-switches and measurements. Their primary task is to protect the power network and perform the control operations either launched from their front panel or from the power distribution system level. Additionally these devices collect the information from the power network related to the faults and power quality harmonics [2]. These devices operate and collect information on the high accuracy level (milliseconds). This information is then sent to the power distribution system, such as MicroSCADA, based on the filtering definitions made in the protection & control device configuration phase. For the measurements it is typical that only the values exceeding the configured dead band value in device are sent into system. However, the switch position indications and other process critical state changes (typically binary) are always sent from the device to system level together with proper time stamps. It is because these are also meaningful, when the changes in the process are later investigated from the system history archive [3][4]. Additional Distribution Management System (DMS) contains the accurate information on the physical power network (location of wooden poles) and customers supplied by specific feeders. This information is then utilized, when DMS participates into fault location process and provides the suggestion of restoration operations to the operator. This functionality is based on the

MicroSCADA and DMS conversation, in which also the information is fetched from the protection & control devices by MicroSCADA.

However, because the functionality of the existing power distribution systems consists of more or less closed software components used in these systems, it means that these components has to be extended to expose their information in a way that it becomes applicable for the semantic web technologies. In the long run the functionality of system using semantic web based solution could be easily expanded and integrated with other systems. It means the system becomes more flexible and adjustable.

The primary requirement for the operator dashboard is that it can be used for monitoring direction, i.e. the intention is to provide assistance to the human operator towards power network operations. Whereas the secondary requirement for the dashboard is that it can be used for the controlling purposes as well. The history of controlling operations made by human operator is archived with the related contextual information of the encountered situations.

The operator dashboard should consist of the information provided by the sources such as protection and control devices, power distribution systems and distribution management system. The intention is not to re-invent the wheel again related to the functionality and algorithms implemented into these existing components. However to combine this functionality over the LAN (Local Area Network) or WAN (Wide Area Network) in the form of external resources, requires that these existing components are extended to be recognized as web services. The information from these extended components should be delivered to the operator dashboard in the case of pre-fault situation or when the fault has already occurred in the power network.

Additionally electrical network operator (expert) could use sources of additional information about power network-context information (e.g. geographical positioning of the power network node, weather condition, etc.) [5]. The context information support use in making decision about fault. So the proposed framework in the paper is going to describe the architecture of the framework, which allows integrating heterogeneous resources and using context information in the process of fault identification and location.

Information about faults on the similar circumstances including human operator decisions are re-used for the similar situations. Prerequisite for this is that the experience and data have been distributed among network nodes. In the case of re-structuring occurs in LAN or WAN, the physical location of network nodes may change.

5.2 General architecture

The architecture of platform should include existing systems and components and integrate it with new solutions. The architecture of the multiagent environment presented on the Figure 9. According to the GUN vision [9,12] the architecture of the future power network infrastructure could be found in Figure 9. All resources are separated to tree types:

- Device;
- Expert;
- Web service (External resource).

Each resource of the platform has a related software agent. The agent enhances functionality of the component. The information about electric device(feeder) is monitoring by the device agent [6]. The expert agent displays information for diagnosis from the device to the human readable format. The web services provides additional (context related information for the system) and algorithms for fault identification and location. Each component has local

storage. The storage contains ontology and parameters and fault diagnosis information. Joseki RDF storage has been used as storage in the software implementation. It's open source tool produced by Hewlett Packard.

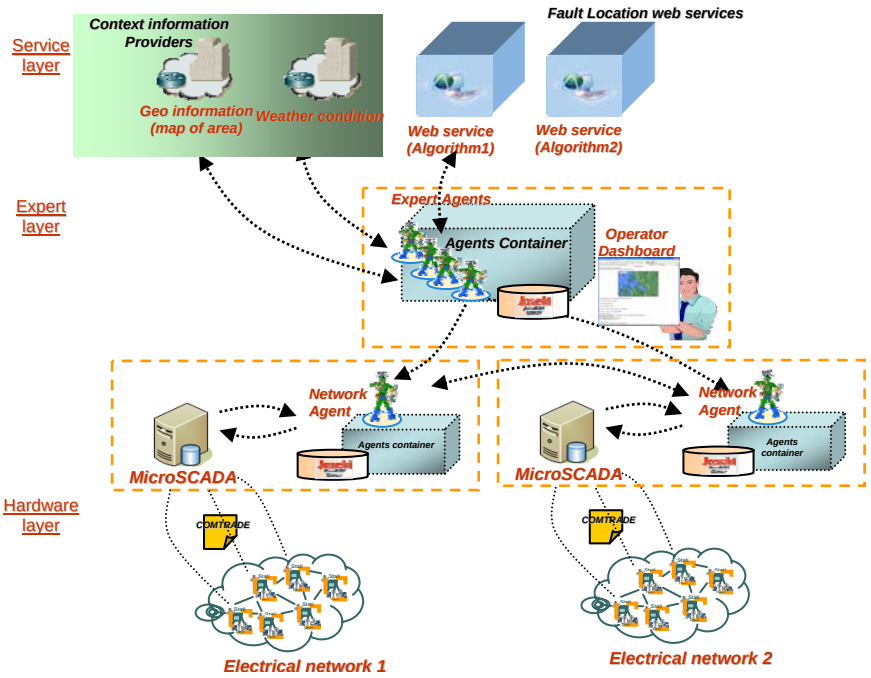

Fig. 9. Architecture of multi-agent electrical distribution controlling environment

5.2.1 Data format

The data to the system from the electric device comes in COMTRADE format[10,11]. COMTRADE is a format of data supported by electrical engineering society. The IEEE Common Format for Transient Data Exchange (COMTRADE) standard specifies a file format for high-speed oscillographic data on multiple data channels and is typically used to capture detailed waveform data of power system events.

The data in COMTRADE format presented as a set of files:

- description file (.INF),
- configuration file (.CFG),
- data file (.DAT).

The interaction between hardware components (feeders) and software components are based on the FTP protocol. The data comes from feeder to SCADA (System Control and Data Acquisition) system.

The agent is able to access functionality of microSCADA system via JavaAPI. The agent extracts data from the microSCADA transforms it to the semantically rich format, store it locally and send it to the expert agent.

In order to be used by external application data should be transformed into the semantically rich format. The transformation is realized by the device agent. The information transforms according to the ontology stored on the local Joseki storage. The outcome of this transformation is data in an RDF/XML format.

5.2.2 Integration with Web Services

Information coming from the device agent should be integrated with context information and shown to the expert. The context information could come from the external web services. The web services are able to provide such information as weather forecast, information about ongoing forest work, etc. In the implemented software system geographical information has been extracted from external web service[13]. The information coming from device agent is integrated with map information and description of the network. As an example of web service we have chosen Google map web service. Google map is providing geographical information according to the provided altitude and longitude. The architecture of the interaction between system components is presented in Figure 10. The agent executes when it gets information from the device agent.

One more way to use advantages of web service is providing access to the various fault detection and location algorithms. The payment for using web service could be subscription fee(month, year) or fee per transaction.

All algorithms could be accessible on-line and experts could easily use it in case if set of local algorithms is not enough. The web service architecture could be used by the customers, who has small electricity power network. So this type of users has no need to buy expensive management systems, such as DMS. They could use functionality the DMS online. The set of available algorithms could be expanded. There is a set of fault detection and location algorithms already implemented in DMS system.

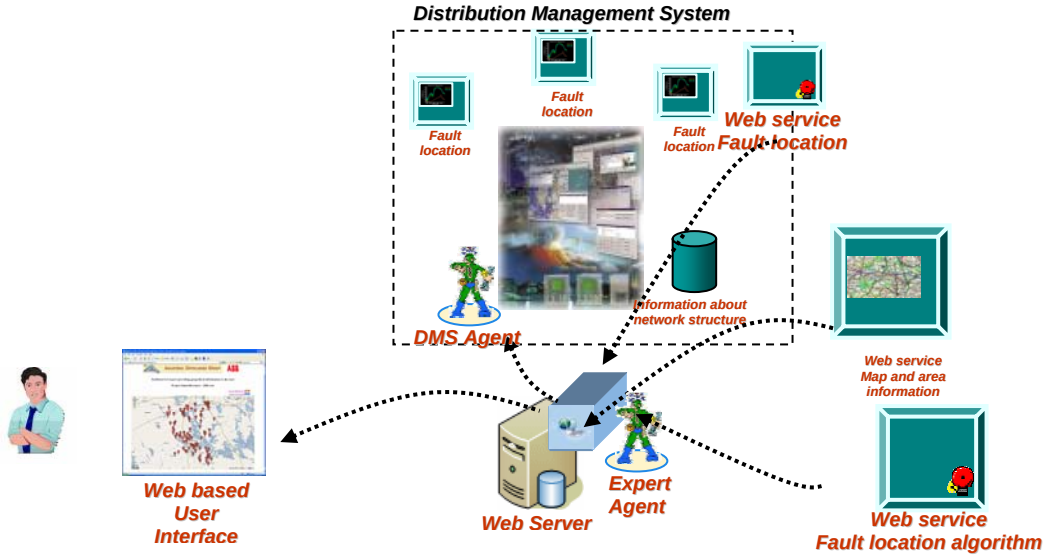

Fig. 10 Example of integration external web service and DMS system

Unfortunately, current version of DMS doesn't have any programming interface to use functionality of the system by external applications. Currently, there is possibility to extract data just from database of DMS. This database contains structure of the electricity network and description of each node. The information in database presented in internal format, which impossible to translate.

5.3 Software development

In order to demonstrate the architecture, a demonstration environment was developed. The demonstration environment of agent-based platform in Power Network Management field was developed using facilities of Technobotnia research lab. The Facilities of research lab

includes power network simulator, automatic relay(for control of electricity network) and microSCADA system(monitor and control electricity network). The view of power network testbed presented in the Figure 11.

Fig. 11. The view of power network testbed

On the right side of power network testbed situated set of switches and breakers. The work and faults in the power network could be emulated using switches and breakers.The automatic relay situated on the left side of testbed allowed to monitor the values of electrical power network parameters. In case of fault situation relay should make dead the line, where the fault happened.

One of the main difficulties was in configuration microSCADA system for getting online information about electricity network parameters from automatic relay. The simulator mainly was used for learning purposes. MicroSCADA was not configured properly. It was connected to power network simulator just partially. The configuration of microSCADA allowed controlling several switches and getting information from automated relay just during start up. In process of configuration were involved a lot of people from ABB side.

5.3.1 Setup and configuration

The microSCADA allows to control and to monitor the status of the electrical power network. MicroSCADA provides a various set of tools for representation information about power network status.

All information in microSCADA represented as objects. The information from about online observed parameters could be read from Process_Object. The history of the parameter could be observed from Data_Object. The event object could invoke specified procedure, when some event happened. Time channels allow executing of specified procedure after some period of time. In order to get information about network parameter the value of corresponding object should be read.

The Object Navigator tool allows as to view and to modify the objects in microSCADA. The screenshot of Object Navigator tool could be found from Figure 12.

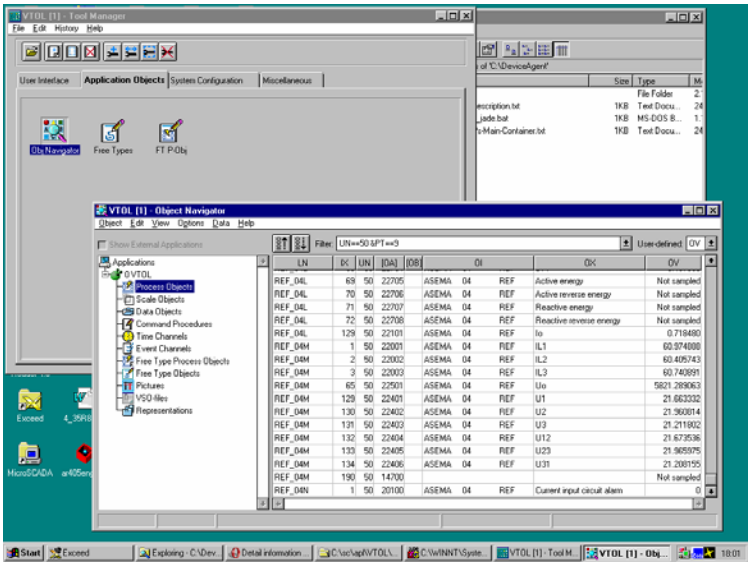

Fig. 12. The window of Object Navigator tool of microSCADA

The process of configuration includes creation set of process objects, for monitoring each of the parameters. It was created six new process objects for monitoring current (I1, I2, I3) and voltage(U1,U2,U3) each phase of three phase feeder. The information about each parameter stores in corresponding Process_Objects. The information store in process objects each three seconds, by invoking defined time channel.

In order to get online values from relay have been done several configuration actions. The first stage configuration was done on the automatic relay. The frequency of updating information and address of the event were specified on the relay. The rest of the configuration was done in microSCADA. System Configuration Tool was used as a tool for configuration. The view of the System Configuration tool could be found on Figure 13.

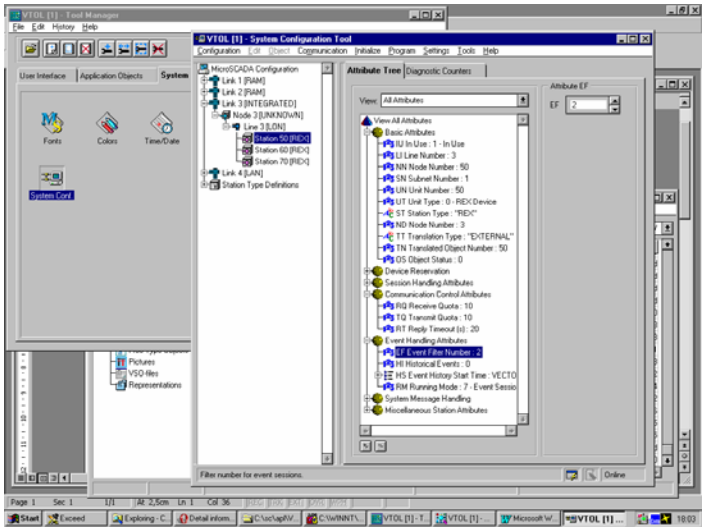

Fig. 13. Window of System Configuration Tool

The functionality of the system could be easily expanded. The system has internal language, which allows to develop various procedure(scripts) and user interface application. The name of the language is SCIL. Using Test dialog tool SCIL applications could be tested and run. One more function of the Test dialog is to provide command line interface for observing values of objects. The screenshot of the Test dialog tool is presented on Figure 14.

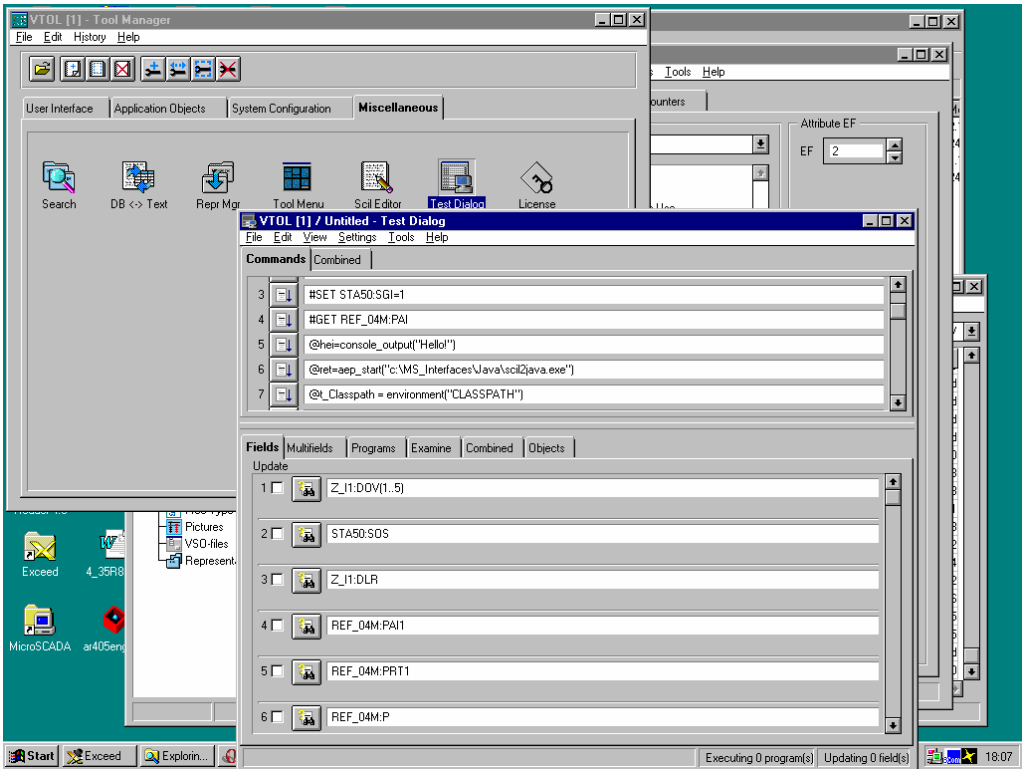

Fig. 14. Screenshot of Test dialog tool

During process of configuration SCIL programm was designed. This program allows to intercept and to display in microSCADA console window messages send by relay. The fragment of code could be found below in Figure 15.

```
; Program starts
#IF %UN == 50 #THEN #BLOCK
  @i_Message = CONSOLE_OUTPUT("UN = 'UN', OA = 'OA'")
#BLOCK_END
; Program ends
```

Fig.15. Example of code for interception messages from automatic relay and display it in console window.

5.3.2 Description of the demonstration environment

The software demonstration environment is to demonstrate integration and representation information from heterogeneous sources using agent-based technology. All collected information presented on Expert Dashboard. Expert Dashboard provides information about power network. The dashboard integrates information from different sources and displays it via human friendly interface. Expert could get such information as location of the electrical network and detail information about it. All presented information could be divided for two parts: electrical network information and context information. Electrical network information presents information about status of the power network. The context information stands for providing additional information to the Expert. Base on the context information expert could make more accurate decision. The information comes from heterogeneous system and

application. Some information is extracted from electrical distribution software applications (microSCADA). The rest comes from web - services(DMS).

Displaying information includes:

- information from microSCADA system(online monitoring of power network)
- information from DMS(description of the power network)
- map of area, where network located
- weather information
- different sort of media information(picture, video, sound)

All users of the dashboard should get possibility to add new information to the map. The users should have possibility to specify new marker on the map and provide annotation(text, link). In order to make system automated and easy for integration the agent technology should be used.

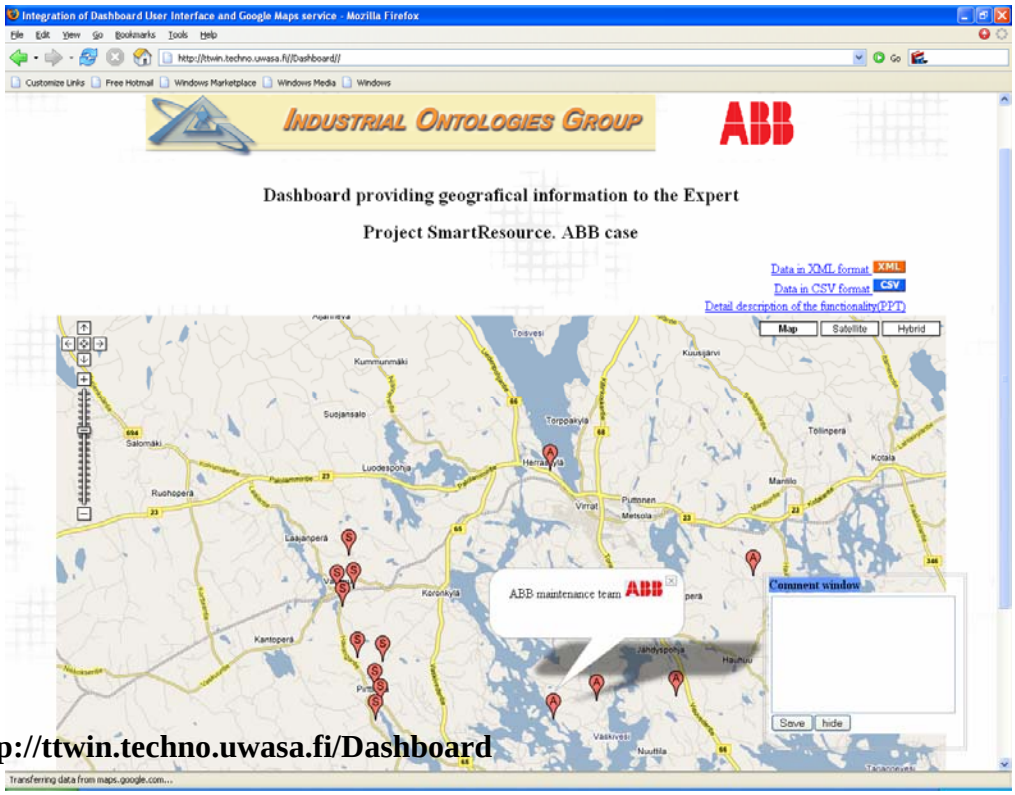

Fig. 16. Screenshot of web-based demonstration environment. Expert's dashboard

Device agent. The monitoring of the electrical power network nodes is realized by device agent. Device agent situated on the same platform as microSCADA. Agent and microSCADA interact with each other via JavaAPI. Agent accesses data about current parameters in electrical network from microSCADA system. Base on easy algorithms, agent analyses data and make decision about status of the electrical network. In case if some fault detected Device agent informs Dashboard agent about fault. In case if fault is not detected agent keeps monitoring network.

DMS Agent. DMS system is stand for proving support and additional information about power network. DMS includes several faults detection and location algorithms. With the

purpose of using functionality of DMS system the DMS agent has been used. DMS agent allows to extract information from DMS and to use built-in algorithms.
Dashboard Agent. The purpose of DMS agent are to collect and integration information from web services and agents dashboard. Agent should present all collected information in web-based user friendly interface.

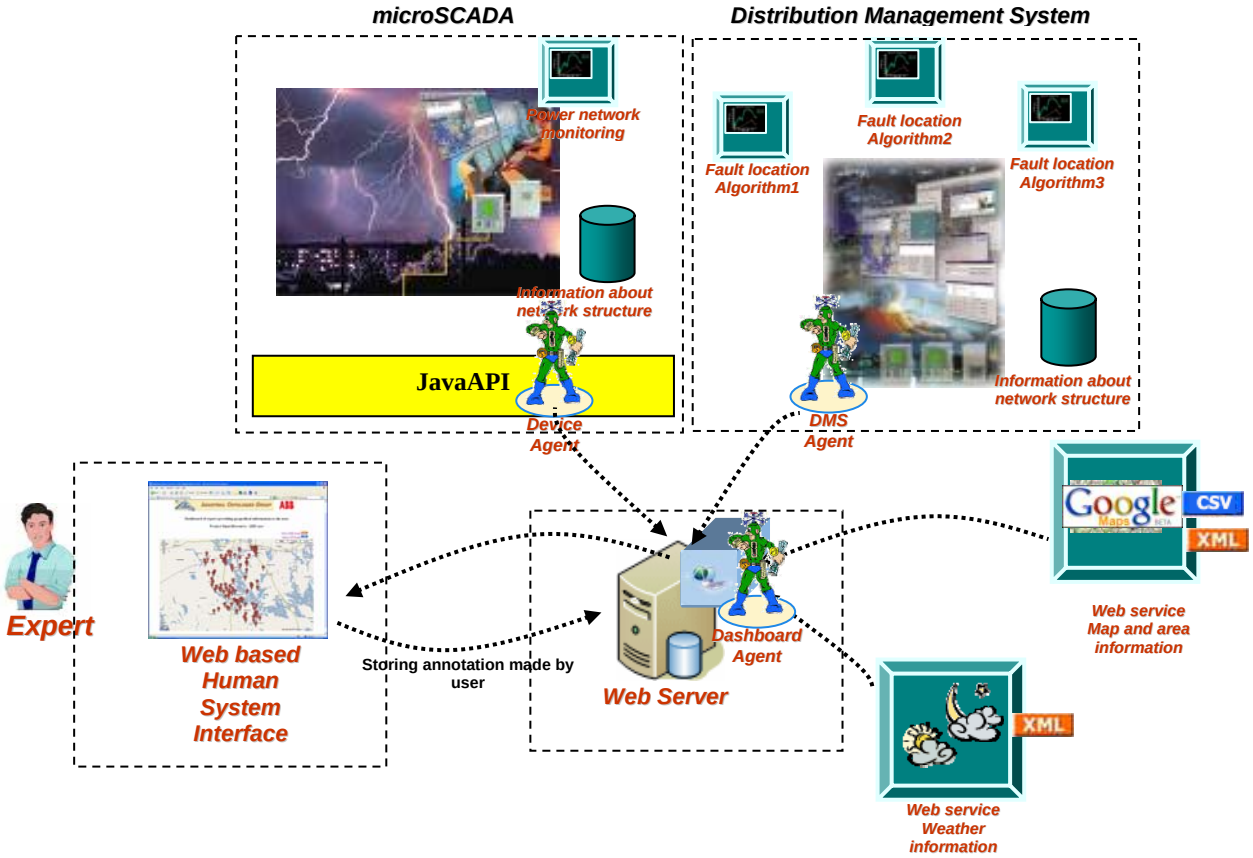

Fig.17. Architecture of the demonstration environment

5.3.3 Performed experiments

Developed software was connected to microSCADA system. The agents were running on different computers. The simulator of the power network allowed us to imitate the fault in the electricity power network. The simulator allows imitating such electrical power network faults as:

- Two-phase short circuit
- Three-phase short circuit
- Earth fault

The purpose of the agents was monitor and informing expert about fault in the network. The Device agent collects value from electricity network using microSCADA and JavaAPI. This data sent to the dashboard agent. DashboardAgent processes all data and display it to the expert. The data from the microSCADA displays to the expert as a set of charts. Each chart shows the values of each parameter. Example of charts could be found from Figure 18.

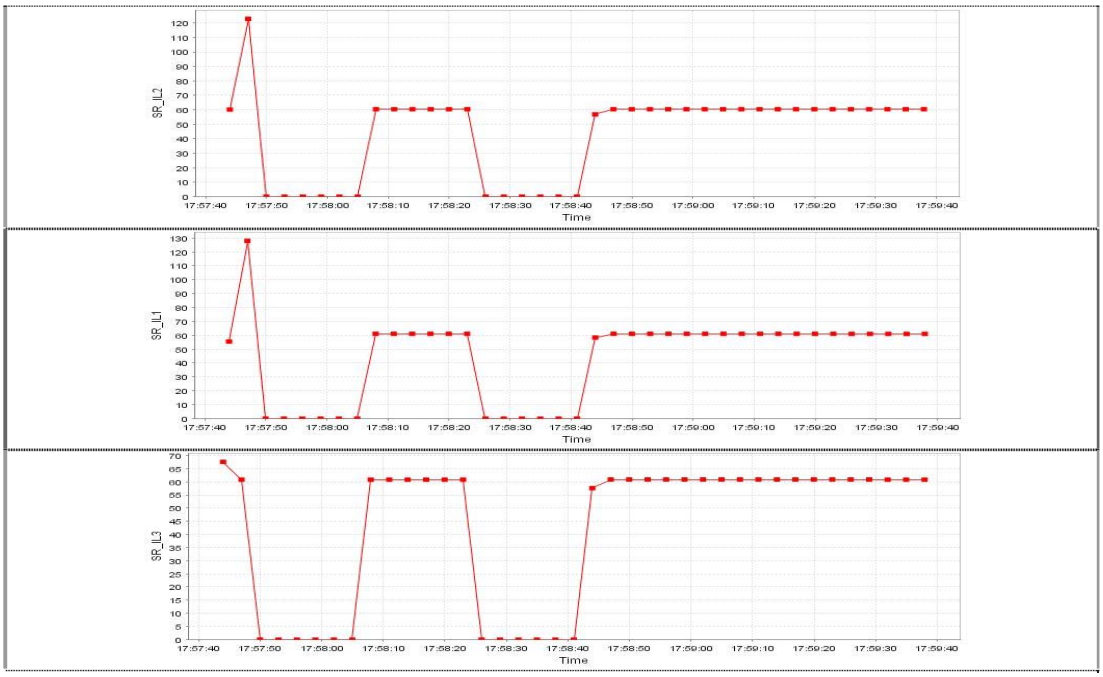

Fig.18. Dashboard screenshot during imitation of different faults

In the Figure 18, you can see the results of experiments, which was done in Technobothnia research lab. After running simulation the Expert's dashboard notifies expert about two faults, which happened recently. The first fault was two phase short cycle. The short cycle happened on phase 1 and 2. This fault was identified by rapid jump and death current on phase 1 and 2. The second fault happened in 18 seconds then the first fault was fixed. The second fault is three phase short cycle. The fault was detected by rapid death the current in all three phases.

6 Conclusions

This report described a case study in the domain of distributed power network maintenance we have been performing in collaboration with ABB Company (Distribution Automation unit). The goal was to study the potential add-value which ABB could receive from introducing Semantic Web technologies, and Global Understanding Environment (GUN) framework in particular, into their business. Development of a prototype, for demonstration of the concept purposes, was a part of the study as well. The description of the practical work on adaptation of the general SmartResource platform for the electricity distribution field is a part of this report too.

An important feature of described solutions is that the existing software systems are not supposed to be replaced. Utilization of the SmartResource platform should aim at extending the interactions of existing systems and integration with various external systems, data storages, information services, and algorithms, which can be found in other organizations, or on the Internet (Figure 19).

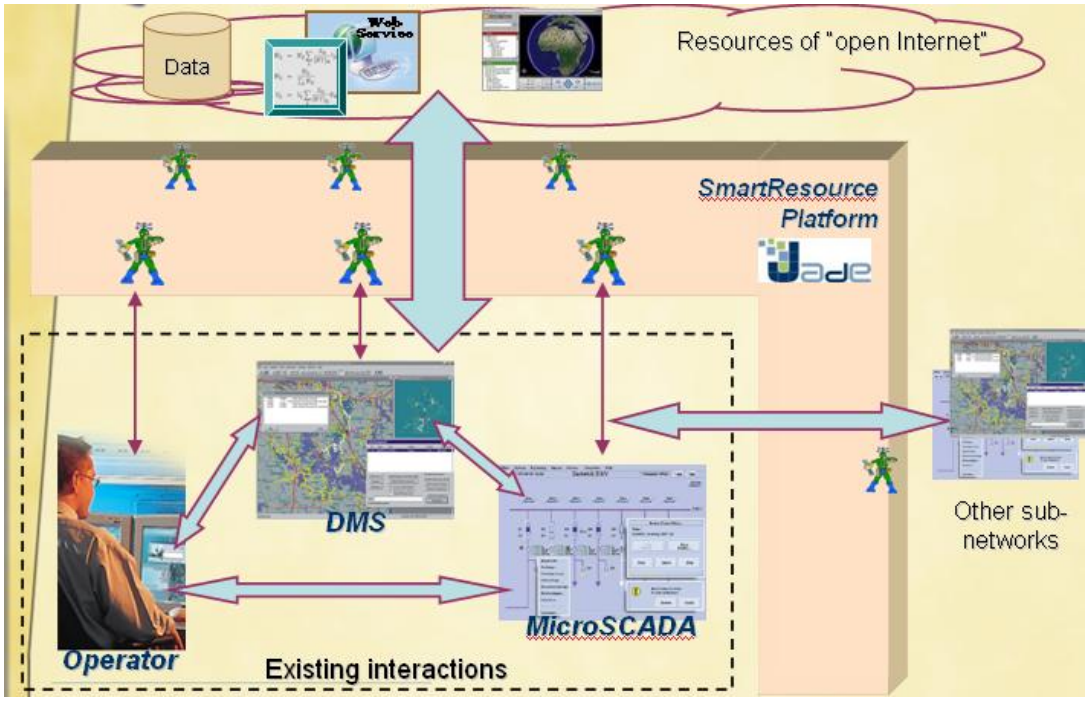

Fig.19. Extending existing interactions

In process of work, we discovered though several problems in existing ABB software products, hindering such development. A major of them is that DMS system does not provide any application programming interface (API), which would allow accessing its useful functionality by external applications, not only a human operator.

7 References

1. Y. Tomita, C. Fukui, H. Kudo, J. Koda, K. Yabe, "A Cooperative Protection System with an Agent Model", IEEE Transactions on Power Delivery, Vol.13, no 4, October 1998, pp. 1060- 66.
2. D. V. Coury, J. S. Thorp, K. M. Hopkinson, K. P. Birman "Improving the protection of EHV teed feeders using local agents"
3. Wong SK, Kalam A (1997) An Agent Approach to Designing Protection Systems. Presented at IEE Conference Publication
4. Hines, Paul, H. Liao, D. Jia, and S. Talukdar. "Autonomous Agents and Cooperation for the Control of Cascading Failures in Electric Grids." Proc. of the IEEE Conf. on Networking, Sensing, and Control. Tuscan, Mar. 2005.
5. GIS integration with SCADA, DMS & AMR in Electrical Utility. http://www.gisdevelopment.net/proceedings/mapindia/2006/energy/mi06ene_184.htm
6. M.Kuzunovic, Fellow IEEE, G.Latisko "Automated monitoring functions for improved power system operation and control"
7. Java Agent Development framework JADE homepage http://jade.tilab.com/
8. Hyacinth S. Nwana and Divine T. Ndumu "A Perspective on Software Agents Research"
9. Kaykova O., Kononenko O., Terziyan V., Zharko A., Community Formation Scenarios in Peer-to-Peer Web Service Environments, In: Proceedings of the IASTED International Conference on Databases and Applications (DBA 2004), Innsbruck, Austria, 17-19 February, 2004, ACTA Press, ISBN: 0-88986-383-0, ISSN: 1027-2666, pp. 62-67.
10. L. Ippolito and P. Siano "An Agent based System for Electrical Components Protection" Italy
11. ECStd. 600255-24, "Common format for transient data exchange(COMTRADE)for power system", first edition 2001-05 International Electrotechnical Commission,2001
12. Kaykova O., Khriyenko O., Kovtun D., Naumenko A., Terziyan V., Zharko A., General Adaption Framework: Enabling Interoperability for Industrial Web Resources, In: International Journal on Semantic Web and Information Systems, Idea Group, ISSN: 1552-6283, Vol.1, No. 3, July-September 2005, pp. 31-63. RDF standard
13. Demonstration of the Integration of Dashboard User Interface and Google Maps service http://ttwin.techno.uwasa.fi/Dashboard