Global Understanding Environment: Applying Semantic and Agent Technologies to Industrial Automation

Vagan Terziyan, Artem Katasonov

Industrial Ontologies Group, Agora Center, University of Jyväskylä,

P.O. Box 35, 40014 Jyväskylä, Finland

e-mail: vagan@jyu.fi, artem.katasonov@jyu.fi

phone: +358 14 260 4618, +358 14 260 2769

fax: +358 14 260 4981

Global Understanding Environment: Applying Semantic and Agent Technologies to Industrial Automation

Industry pushes a new type of Internet characterized as the Internet of Things, which represents a fusion of the physical and digital worlds. The technology of the Internet of Things opens new horizons for industrial automation, i.e. automated monitoring, control, maintenance planning, etc., of industrial resources and processes. Internet of Things definitely needs explicit semantics, even more than the traditional Web – for automatic discovery and interoperability among heterogeneous devices and also to facilitate the behavioral coordination of the components of complex physical-digital systems. In this chapter, we describe our work towards the Global Understanding Environment (GUN), a general middleware framework aimed at providing means for building complex industrial systems consisting of components of different nature, based on the semantic and the agent technologies. We present the general idea and some emergent issues of GUN and describe the current state of the GUN realization in the UBIWARE platform. As a specific concrete case, we use the domain of distributed power network maintenance. In collaboration with the ABB Company we have developed a simple prototype and a vision of potential add-value this domain could receive from introducing semantic and agent technologies, and GUN framework in particular.

Keywords: Internet-Based Technology, Middleware, Ontologies, Semantic Data Model, Software Agents, Industrial Automation, Heterogeneous Resources, Interoperability

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 also opens 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" (Kephart and Chess, 2003) in which human operators will be unable to *manage* the complexity of interactions, neither even architects will be able to *anticipate* this complexity and thus *design* the systems effectively. 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 vision of autonomic computing (e.g. Kephart and Chess, 2003) 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. According to this vision, the self-manageability of a complex system requires its components to be to a certain degree autonomous themselves. Therefore, 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 (see Section 2).

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. The semantic 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* (Brock and Schuster, 2006). Our work subscribes to this view. Moreover, we believe that the semantic technologies can facilitate not only the discovery of heterogeneous components and data integration, but also the behavioral coordination of those components (see Section 2).

In this paper, we describe our work on the *Global Understanding Environment (GUN)* (the concept introduced in Terziyan, 2003, 2005). This work is conducted in the line of projects of the Industrial Ontologies Group at the University of Jyväskylä including SmartResource (2004-2007, see http://www.cs.jyu.fi/ai/OntoGroup/SmartResource_details.htm) and ongoing UBIWARE (Smart Semantic Middleware for Ubiquitous Computing, 2007-2010, see http://www.cs.jyu.fi/ai/OntoGroup/UBIWARE_details.htm). GUN is a general middleware framework aiming at providing means for building complex industrial systems consisting of components of *different* nature, based on the semantic and agent technologies. A very general view on GUN is presented in Figure 1; a description of GUN will be given in Section 3.

When applying the semantic approach in the domain of industrial automation, it should be obvious that the semantic technology has to be able to describe resources not only as passive functional or non-functional entities, but also to describe their behavior (proactivity, communication, and coordination). In this sense, 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. From the Semantic Web point of view, GUN could probably be referred to as *Proactive, Self-Managed Semantic Web of Things*. We believe that such Proactive Self-Managed Semantic Web of Things can be the future "killer application" for the Semantic Web.

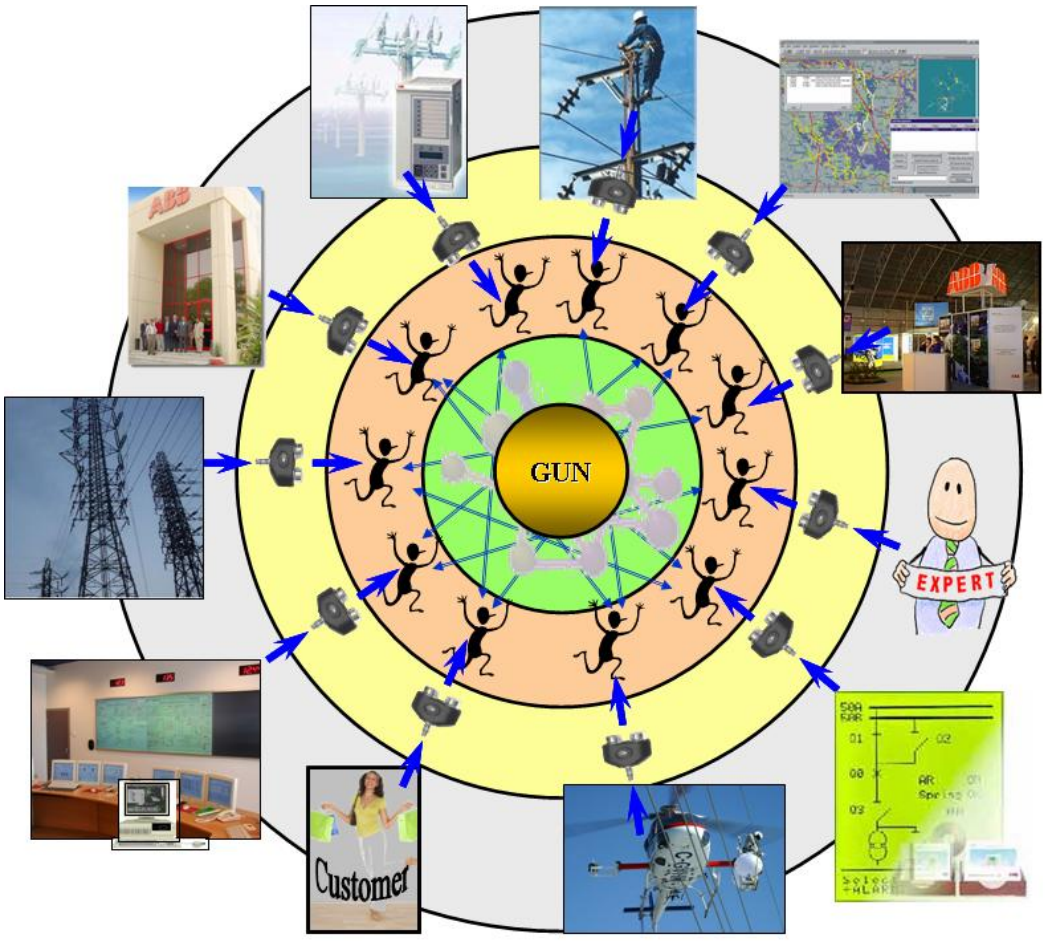

Fig. 1. The Global Understanding Environment

As a specific concrete case for this paper, we use the domain of distributed power network maintenance. We describe our existing prototype and the vision we developed in collaboration with ABB Company (Distribution Automation unit).

The further text is organized as follows. In Section 2, we discuss the background for GUN and comment on related research. In Section 3, we present the general idea of GUN, provide references to more detailed information on already elaborated parts of it, and further focus on some recent issues in our work. In Section 4, we describe the achieved state of the GUN realization in the *UBIWARE Platform*. Section 5 presents the case study from the domain of distributed power network maintenance. Finally, Section 6 presents discussion and future work directions.

BACKGROUND AND RELATED RESEARCH

Semantic Technologies for the Internet of Things

An excellent analysis of the today's status and the roadmap for the future development of the Internet of Things has been made as collective effort of academy and industry during the conference organized by DG Information Society and Media, Networks and Communication Technologies Directorate in Brussels (Buckley, 2006). It was pointed out that the Internet of Things characterizes the way that information and communication technologies will develop over the next decade or so. The Internet of Things represents a fusion of the physical and digital worlds. It creates a map of the real world within the digital world. The computer's view of the physical world may, depending on the characteristics of sensor network, possess a high temporal

and spatial resolution. The Internet of Things may react autonomously to the real world. A computer's view of the world allows it to interact with the physical world and influence it. The Internet of Things is not merely a tool to extend the human capability. It becomes part of the environment in which humans live and work, and in doing that it can create an economically, socially and personally better environment. In industry and commerce, the Internet of Things may bring a change of business processes (Buckley, 2006).

According to Buckley (2006), the devices on the Internet of Things will have several degrees of sophistication and the final one makes *Proactive Computing* (INTEL terminology) possible. These devices (sometimes called *Smart Devices*) are aware of their context in the physical world and able to react to it, which may cause the context to change. The power of the Internet of Things and relevant applications arises because devices are interconnected and appropriate service platforms are emerging. Such platforms must evolve beyond the current limitations of static service configurations and to move towards service-oriented architectures. Interoperability requires that clients of services know the features offered by service providers beforehand and semantic modeling should make it possible for service requestors to understand what service providers have to offer. This is a key issue for moving towards an open-world approach where new or modified devices and services may appear at any time. This also has implications on requirements for middleware, as these are needed to interface between the devices that may be seen as services, and applications. This is a key issue to progress towards device networks capable of dynamically adapting to context changes as may be imposed by application scenarios (e.g. moving from monitoring mode to alarm mode and then to alert mode may imply different services and application behaviors). Devices in the Internet of Things might need to be able to communicate with other devices anywhere in the world. This implies a need for a naming and

addressing scheme, and means of search and discovery. The fact that devices may be related to an identity (through naming and addressing) raises in turn a number of privacy and security challenges. A consistent set of middleware offering application programming interfaces, communications and other services to applications will simplify the creation of services and applications. Service approaches need to move from a static programmable approach towards a configurable and dynamic composition capability.

In Lassila and Adler (2003), the ubiquitous computing is presented as an emerging paradigm qualitatively different from existing personal computing scenarios by involving dozens and hundreds of devices (sensors, external input and output devices, remotely controlled appliances, etc). A vision was presented for a class of devices, so called *Semantic Gadgets*, which will be able to combine functions of several portable devices users have today. Semantic Gadgets will be able to automatically configure themselves in new environments and to combine information and functionality from local and remote sources. Semantic Gadgets should be capable of semantic discovery and device coalition formation: the goal should be to accomplish discovery and configuration of new devices without a human in the loop. Authors pointed out that critical to the success of this is the existence or emergence of certain infrastructures, such as the World Wide Web as a ubiquitous source of information and services and the Semantic Web as a more machine- and automation-friendly form of the Web.

Later, Lassila (2005a, 2005b) discussed possible application of semantic technologies to mobile and ubiquitous computing arguing that ubiquitous computing represents the ultimate "interoperability nightmare". This application is motivated by the need for better automation of user's tasks by improving the interoperability between systems, applications, and information. Ultimately, one of the most important components of the realization of the Semantic Web is

"serendipitous interoperability", the ability of software systems to discover and utilize services they have not seen before, and that were not considered when and where the systems were designed. To realize this, qualitatively stronger means of representing service semantics are required, enabling fully automated discovery and invocation, and complete removal of unnecessary interaction with human users. Avoiding a priori commitments about how devices are to interact with one another will improve interoperability and will thus make dynamic, unchoreographed ubiquitous computing scenarios more realistic. The semantic technologies are qualitatively stronger approach to interoperability than contemporary standards-based approaches.

To be truly pervasive, the devices in a ubiquitous computing environment have to be able to form a coalition without human intervention. In Qasem et al. (2004), it is noticed that ordinary AI planning for coalition formation will be difficult because a planning agent cannot make a closed-world assumption in such environments. Agent never knows when e.g. it has gathered all relevant information or when additional searches may be redundant. Local closed-world reasoning has been incorporated in Qasem et al. (2004) to compose Semantic Web services and to control the search process. The approach has two main components. The first is Plan Generator, which generates a plan that represents a service composition. The second component, the Semantic Web mediator, provides an interface to the information sources, which are devices in the ubiquitous computing environments.

The advances around the Semantic Web and Semantic Web services allow machines to help people to get fully automated anytime and anywhere assistance. However, most of the available applications and services depend on synchronous communication links between consumers and providers. In Krummenacher and Strang (2005), a combination of space-based computing and

Semantic Web named as *semantic spaces* is introduced to provide a communication paradigm for ubiquitous services. The semantic spaces approach introduces a new communication platform that provides persistent and asynchronous dissemination of machine-understandable information, especially suitable for distributed services. Semantic spaces provide emerging Semantic Web services and Semantic Gadgets with asynchronous and anonymous communication means. Distributing the space among various devices allows anytime, anywhere access to a virtual information space even in highly dynamic and weakly connected systems. To handle all the semantic data emerging in such systems, data stores will have to deal with millions of triples. In consequence reasoning and processing the data becomes highly time and resource consuming. The solution is to distribute the storage and computation among the involved devices. Every interaction partner provides parts of the space infrastructure and data.

One question is whether Semantic Web is ready to provide services, which fit the requirements of the future Internet of Things? The original idea of Semantic Web (Berners-Lee *et al.*, 2001) is to make Web content suitable not only for human browsing but also for automated processing, integration, and reuse across heterogeneous applications. The effort of the Semantic Web community to apply its semantic techniques in open, distributed and heterogeneous Web environments have paid off: the Semantic Web is evolving towards a real Semantic Web (Sabou *et al.*, 2006). Not only the number of developed ontologies is dramatically increasing, but also the way that ontologies are published and used has changed. We see a shift away from first generation Semantic Web applications, towards a new generation of applications, designed to exploit the large amounts of heterogeneous semantic markup, which are increasingly becoming available. In Motta and Sabou (2006), a number of criteria are given, which Semantic Web

applications have to satisfy on their move away from conventional semantic systems towards a new generation of Semantic Web applications:

- *Semantic data generation vs. reuse* - the ability to operate with the semantic data that already exist, i.e. to exploit available semantic markup.

- *Single-ontology vs. multi-ontology systems* - the ability to operate with huge amounts of heterogeneous data, which could be defined in terms of many different ontologies and may need to be combined to answer specific queries.

- *Openness with respect to semantic resources* - the ability to make use of additional, heterogeneous semantic data, at the request of their user.

- *Scale as important as data quality* - the ability to explore, integrate, reason and exploit large amounts of heterogeneous semantic data, generated from a variety of distributed Web sources.

- *Openness with respect to Web (non-semantic) resources* - the ability to take into account the high degree of change of the conventional Web and provide data acquisition facilities for the extraction of data from arbitrary Web sources.

- *Compliance with the Web 2.0 paradigm* - the ability to enable *Collective Intelligence* based on massively distributed information publishing and annotation initiatives by providing mechanisms for users to add and annotate data, allowing distributed semantic annotations and deeper integration of ontologies.

- *Open to services* - the ability of applications to integrate Web-service technology in applications architecture.

In a nutshell, next generation Semantic Web systems will necessarily have to deal with the increased heterogeneity of semantic sources (Motta and Sabou, 2006), which partly corresponds to the trends related to the Internet of Things roadmap for the future development (Buckley, 2006).

As discussed above, ubiquitous computing systems need explicit semantics for automatic discovery and interoperability among heterogeneous devices. Moreover, it seems that that the traditional Web as such is not enough to motivate the need for the explicit semantics, and this may be a major reason why no killer application for the Semantic technologies has been found yet. In other words, it is not only that the ubiquitous computing needs semantics, but also the Semantic Web may need the emergence of really ubiquitous computing to finally find its killer application. Recently, the US Directorate for Computer and Information Science and Engineering (CISE) and National Science Foundation (NSF) has announced an initiative called Global Environment for Networking Innovations (GENI, http://www.nsf.gov/cise/cns/geni/) to explore new networking capabilities and move towards the *Future Internet*. Some of GENI challenges are: support for pervasive computing, bridging physical and cyberspace with the impact to access the information about physical world in real time, and enabling exciting new services and applications (Freeman, 2006). If the Future Internet will allow more natural integration of sensor networks with the rest of the Internet, as GENI envisions, the amount and heterogeneity of resources in the Web will grow dramatically and without their ontological classification and (semi- or fully-automated) semantic annotation processes the automatic discovery will be impossible.

Semantic Technologies for Inter-Agent Coordination

When it comes to developing complex, distributed software-based systems, the *agent-based approach* was advocated to be a well suited one (Jennings, 2001). From the implementation point of view, agents are a next step in the evolution of software engineering approaches and programming languages, the step following the trend towards increasing degrees of localization and encapsulation in the basic building blocks of the programming models (Jennings, 2000). After the *structures*, e.g., in C (localizing data), and *objects*, e.g., in C++ and Java (localizing, in addition, code, i.e. an entity's behavior), agents follow by localizing their *purpose*, the thread of control and action selection. An agent is commonly defined as an encapsulated computer system situated in some environment and capable of flexible, autonomous action in that environment in order to meet its design objectives (Wooldridge, 1997).

However, the actual benefit of the agent-oriented approach arises from the fact that the notion of an agent is also appropriate as a basis for the analysis of the problem to be solved by the system developed. Many processes in the world can be conceptualized using an agent metaphor; the result of such a conceptualization is either a single agent (or cognitive) description or a multi-agent (or social) description (Bosse and Treur, 2006). Jennings (2001) argued that agent-oriented decompositions (according to the purpose of elements) are an effective way of partitioning the problem space of a complex system, that the key abstractions of the agent-oriented mindset are a natural means of modeling complex systems, and that the agent-oriented philosophy for modeling and managing organizational relationships is appropriate for dealing with the dependencies and interactions that exist in complex systems.

The problem of crossing the boundary from the domain (problem) world to the machine (solution) world is widely recognized as a major issue in software and systems engineering. Therefore, when it comes to designing software, the most powerful abstractions are those that minimize the semantic distance between the units of analysis that are intuitively used to conceptualize the problem and the constructs present in the solution paradigm (Jennings, 2000). A possibility to have the same concept, i.e. agent, as the central one in both the problem analysis and the solution design and implementation can make it much easier to design a good solution and to handle complexity. In contrast, e.g. the object-oriented approach has its conceptual basis determined by the underlying machine architecture, i.e. it is founded on implementation-level ontological primitives such as object, method, invocation, etc. Given that the early stages of software development are necessarily based on intentional concepts such as stakeholders, goals, plans, etc, there is an unavoidable gap that needs to be bridged. Bresciani et al. (2004) even claimed that the agent-oriented programming paradigm is *the only* programming paradigm that can gracefully and seamlessly integrate the intentional models of early development phases with implementation and run-time phases. In a sense, agent-oriented approach postpones the transition from the domain concepts to the machine concepts until the stage of the design and implementation of individual agents (given that those are still to be implemented in an object-oriented programming language).

Although the flexibility of agent interactions has many advantages when it comes to *engineering* complex systems, the downside is that it leads to certain *unpredictability in the run time system*; as agents are autonomous, the patterns and the effects of their interactions are uncertain (Jennings, 2000). This raises a need for effective coordination, cooperation, and negotiation mechanism. (Those are in principle distinct, but the word “coordination” is often used as a

general one encompassing all three; so for the sake of brevity we will use it like that too.) Coordination aims at avoiding negative interactions, e.g. two agents simultaneously accessing the same non-shareable resource, as well as exploiting positive interactions, e.g. one agent performs an action and shares its results so that another agent would not need to repeat the same action. Jennings (2000) discussed that it is common in specific systems and applications to circumvent coordination difficulties by using interaction protocols whose properties can be formally analyzed, by adopting rigid and preset organizational structures, and/or by limiting the nature and the scope of the agent interplay. However, Jennings asserted that these restrictions also limit the power of the agent-based approach; thus, *in order to realize its full potential some longer term solutions are required*. Emergence of such a longer term solution that would allow flexible yet predictable operation of agent systems seems to be a prerequisite for wide-scale adoption of the agent-oriented approach.

The available literature sketches two major directions of search for such a longer term solution:

- D1: *Social level* characterization of agent-based systems. E.g. Jennings (2000) stressed the need for a better understanding of the impact of sociality and organizational context on an individual's behavior and of the symbiotic link between the behavior of the individual agents and that of the overall system.

- D2: *Ontological* approaches to coordination. E.g. Tamma et al. (2005) asserted a need for common vocabulary for coordination, with a precise semantics, to enable agents to communicate their intentions with respect to future activities and resource utilization and get them to reason about coordination at run time. Also Jennings et al. (1998) put as an

issue to resolve the question about how to enable individual agents to represent and reason about the actions, plans, and knowledge of other agents to coordinate with them.

Recently, some progress has been made with respect to D1, resulting, e.g., in elaboration of the concept of a *role* that an agent can play in an organization (see below). However, with respect to D2 very little has been done. Bosse and Treur (2006) discussed that the ontological understanding among agents requires sharing the following different types of ontologies: an ontology for internal mental properties of the agent A, MentOnt(A), for properties of the agent's (physical) body, BodyOnt(A), for properties of the (sensory or communication) input, InOnt(A), for properties of the (action or communication) output, OutOnt(A), of the agent, and for properties of the external world, ExtOnt(A). Using this distinction, we could describe the present situation as following. The work on explicitly described ontologies was almost exclusively concerned with ExtOnt(A), i.e. the *domain ontologies*. MentOnt(A) comes for free when adopting a certain agent's internal architecture, such as Beliefs-Desires-Intentions (BDI) (Rao and Georgeff, 1991). Also, the communication parts of InOnt(A) and OutOnt(A) come for free when adopting a certain communication languages, such as FIPA's ACL and SL. However, BodyOnt, i.e. the vocabulary for describing preceptors and actuators that the agent has available, the sensory part of InOnt, i.e. the agent's perception vocabulary, and the action part of OutOnt, e.g. the agent's acting vocabulary, are not usually treated. However, sharing these ontologies is a necessary precondition for agents' awareness of and understanding each other's actions, i.e. for D2. Already referred to article by Tamma et al. (2005) is one of the first endeavors into this direction, which however only introduced and analyzed some of the relevant concepts, such as resource, activity, etc.

In our work, we attempt to provide a solution advancing into both D1 and D2 and somewhat integrating both. The current state, namely the architecture of the UBIWARE Platform, will be described later. Some basic thinking, leading to it, is presented here.

On the landscape of research in agent-based systems, we can identify two somewhat independent streams of research, each with its own limitations. The first stream is the research in multi-agent systems (MAS); the second stream is the research in agents' internal architectures and approaches to implementation.

Researchers in MAS have contributed with, among others, various methodologies for designing MAS, such as Gaia (Wooldridge et al., 2000), TROPOS (Bresciani et al., 2004), and OMNI (Vázquez-Salceda et al., 2005). For example, OMNI (which seems to be the most advanced with respect to D1) elaborates on the organizational context of a MAS, defines the relationship between organizational roles and agents enacting those roles, discusses how organizational norms, values and rules are supposed to govern the organization's behavior and thus to put restrictions on individual agents' behaviors. However, OMNI touches only on a very abstract level the question about how the individual agents will be implemented or even function; the agents are treated as rather atoms. One reason is that it is (reasonably) assumed that the agent organization's designer may have no direct control over the design of individual agents. The organization designer develops the rules to be followed and enforcing policies and entities, such as "police" agents, while development of other agents is done by external people or companies. One of few concrete implementation requirements mentioned in OMNI is that a rule interpreter must be created that any agent entering the organization will incorporate, somehow. The OMNI framework also includes explicitly the ontological dimension, which is restricted, however, to a domain ontology only, and thus does not provide much new with respect to D2.

The other stream of research, on individual agents, has contributed e.g. with well-known BDI architecture, and introduced *agent-oriented programming* (Shoham, 1993) along with several *agent programming languages (APL)* such as AGENT-0 (Shoham, 1993), AgentSpeak(L) (Rao, 1996), 3APL (Dastani et al., 2003) and ALPHA/AFAPL (Collier et al., 2005). All of those are declarative languages, follow the BDI model, and are based on the first order logic of n-ary predicates. For example, an agent program in ALPHA consists of declarations of the beliefs and goals of that agent and declaration of a set of rules, including belief rules (generating new beliefs based on existing ones), reactive rules (invoking some actions immediately) and commitment rules (adopting a commitment to invoke an action). Sensors (perceiving environment and generating new beliefs) and actuators (implementing the actions to be invoked) are then pieces of external code, in Java. As discussed above, the agent-oriented approach postpones the transition from the domain concepts to the machine concepts until the stage of the design and implementation of individual agents. The advantage of using an APL is that the transition is postponed even further, until the implementation of particular sensors and actuators.

This advantage is, however, the only one that is usually considered. In some approaches, it is assumed that the agents can communicate their plans encoded in an APL. But, otherwise, the role of APL code does not go much beyond the development stage. APL code is assumed to be written by the developer of an agent and either compiled into an executable program or interpreted in run-time but remaining an agent's intrinsic and static property. APL code is not assumed to ever come from outside of the agent in run-time, neither shared with other agents in any way.

Such export and sharing of APL code would, however, probably make sense in the light of findings from the field of MAS, and also in the light of D2. Methodologies like OMNI describe

an organizational role with a set of rules, and an APL is a rule-based language. So, using an APL for specifying a role sounds as a natural way to proceed. The difference is that APL code corresponding to a role should naturally be a property of and controlled by the organization, and accessed by the agents' enacting the role potentially even in the run-time. Run-time access would also enable the organization to update the role code if needed. The second natural idea is that the agents may access a role's APL code not only in order to enact that role, but also in order to coordinate with the agents playing that role. As one option, an agent can send to another agent a part of its APL code to communicate its intentions with respect to future activities (so there is no need for a separate content language). As another option, if a role's code is made public inside the organization, the agents may access it in order to understand how to interact with, or what to expect from, an agent playing that role.

However, when thinking about using the existing APLs in such a manner, there are at least two issues:

- The code in an APL is, roughly speaking, a text. However in complex systems, a description of a role may need to include a huge number of rules and also a great number of beliefs representing the knowledge needed for playing the role. Also, in a case of access of the code by agents that are not going to enact this role, it is likely that they may wish to receive only a relevant part of it, not the whole thing. Therefore, a more efficient, e.g. a database-centric, solution is probably required.

- When APL code is provided by an organization to an agent, or shared between agents, mutual understanding of the meaning of the code is obviously required. While using first-order logic as the basis for an APL assures understanding of the semantics of the

rules, the meaning of predicates used in those rules still needs to be consistently understood by all the parties involved. On the other hand, we are unaware of tools allowing unambiguous description of the precise semantics of n-ary predicates: sendsMessage(a, b, c) - who is sending what to who.

To summarize the discussion above, in a nutshell our approach is: Let's treat agent programs as data; data that can be stored into a database, queried for, merged, shared between agents, and so on. As with any data in a distributed computer system, there are problems of other-party understanding the meaning of the data and of machine processibility. Therefore, the utilization of the Semantic Web technology is natural. Semantic Web technology makes the meaning of data as explicit and as unambiguous as possible through the Resource Description Framework (RDF) data model that uses binary predicates only, Universal Resource Identifiers (URI), and ontologies. Ontologies can be explicitly modeled, e.g. using Web Ontology Language (OWL), providing the explicit description of the semantics of predicates and enabling semantic inference. Several RDF databases exist including Sesame, Joseki, and Oracle Spatial RDF.

Therefore, our proposition is to create and use an agent programming language which will be RDF-based. In fact, the Semantic Web community already contributed with several rule-based semantic reasoners, which are in many aspects similar to APLs with the exception that they do not involve acquiring new data, i.e. sensing, or invoking any actions, neither external nor even mental, such as removing data. One example is Tim Berners-Lee's own CWM data processor (http://www.w3.org/2000/10/swap/doc/cwm), another one is Euler inference engine (http://www.agfa.com/w3c/euler/). CWM is a forward-chaining reasoner while Euler is backward-chaining reasoner; however, they are interoperable because both are based on Notation3 (http://www.w3.org/DesignIssues/Notation3.html). Notation3 (or N3) was proposed

by Berners-Lee himself as an alternative to the dominant notation for RDF, which is RDF/XML. N3 is a language which is more compact and probably better readable than RDF/XML, and is also extended to allow greater expressiveness. One feature of N3, which goes beyond the plain RDF, is the concept of formula that allow RDF graphs to be quoted within RDF graphs, e.g. {:room1 :hasTemperature 25} :measuredBy :sensor1. An important convention is that a statement inside a formula is not considered as asserted, i.e., as a general truth. (In a sense, it is a truth inside a context defined by the statement about the formula and the outer formulas.) This is in contrast to the plain RDF where every statement is asserted as a truth.

The Semantic Agent Programming Language (S-APL) is in a sense a hybrid of CWM-like semantic reasoners and ALPHA-like agent programming languages. S-APL uses the data model and the notation of N3. Most of CWM constructs are either directly applicable or have equivalents in S-APL. From the CWM point of view, S-APL extends it with common APL features such as BDI architecture, i.e. ability to describe goals and commitments - data items presence of which lead to some executable behavior, and ability to link to sensors and actuators implemented in a procedural language, namely Java. S-APL also includes several solution-set modifiers and counters which have little meaning in the context of pure reasoners like CWM, but can be found e.g. in the standard RDF query language SPARQL. From APLs point of view, S-APL is a language that provides all the features (and more) of exisitng APLs, while being RDF based and thus providing advantages of semantic data model and reasoning.

S-APL is the core language of the UBIWARE Platform. Although this paper provides a description of the UBIWARE Platform, describing S-APL is beyond the scope of the paper. For a detailed description of S-APL, see Katasonov (2008). Note that in Katasonov and Terziyan

(2007), we described the old version of S-APL. Although being based on same basic ideas, that version used different notation and was rather primitive as the current one.

THE GLOBAL UNDERSTANDING ENVIRONMENT (GUN)

GUN Basics

Global Understanding Environment (GUN) (Terziyan, 2003; Terziyan, 2005; Kaykova et al., 2005a) is a concept denoting next generation of Web-based platforms which will make heterogeneous industrial resources (documents, services, devices, business processes, systems, organizations, human experts, etc.) web-accessible, proactive and cooperative in the sense that they will be able to automatically plan own behavior, monitor and correct own state, communicate and negotiate among themselves depending on their roles in a business process, utilize remote experts, Web-services, software agents and various Web applications. Three fundamentals of such platform are *Interoperability*, *Automation* and *Integration*. Interoperability in GUN requires utilization of Semantic Web standards, RDF-based data, metadata and ontologies and semantic adapters for the resources. Automation in GUN requires proactivity of resources based on applying the agent technologies. Integration in GUN requires ontology-based business process modeling and integration and multi-agent technologies for coordination of business processes over resources.

The main players in GUN are the following resources: service consumers (or components of those), service providers (or components of those), and decision-makers (or components of those). All these resources can be artificial (tangible or intangible) or natural (human or other). It is assumed that the service consumers will be able: (a) to proactively monitor own state over

time and changing context; (b) to discover appropriate decision makers and order from them remote diagnostics of their condition, so that the decision makers can decide what maintenance (treatment) services are applied to that condition; (c) to discover appropriate service providers and order from them the required maintenance.

Main layers of the GUN architecture are shown in Figure 1 (in Section 1). Industrial resources (e.g. devices, experts, software components, etc.) can be linked to the Semantic Web-based environment via adapters (or interfaces), which include, if necessary, sensors with digital output, data structuring (e.g. XML) and semantic adapter components (XML to Semantic Web formats). Agents are assumed to be assigned to each resource and to be able to monitor semantically enriched data coming from the adapter about states of the resource, decide if more deep diagnostics of the state is needed, discover other agents in the environment, which represent decision makers and exchange information (agent-to-agent communication with semantically enriched content language) to get diagnoses and decide if maintenance is needed. It is assumed that decision making Web-services will be implemented based on various machine learning algorithms and will be able to learn based on samples of data taken from various service consumers and labeled by experts. Utilization of the agent technologies within GUN framework allows mobility of service components between various platforms, decentralized service discovery, FIPA communication protocols utilization, and MAS-like integration/composition of services.

The SmartResource project, which was mentioned above, in its research and development efforts analyzed Global Understanding Environment decomposing it into three main parts:

The first is the *General Adaptation Framework (GAF)* for semantic interoperability. GAF has to provide means for semantic description of industrial resources, including dynamic and context-sensitive information. The central part in GAF is played by the Resource State/Condition Description Framework (RscDF). An implementation of GAF for a specific domain is assumed to include also an appropriate RscDF-based domain ontology, an appropriate RscDF Engine and the family of s*emantic adapters to resources* to provide an opportunity to transform data from a variety of possible resource data representation standards and formats to RscDF-based and back. For more details about RscDF and GAF see (Kaykova *et al.*, 2005b) and (Kaykova *et al.*, 2005a).

The second is the *General Proactivity Framework (GPF)* for automation and proactivity. GPF has to provide means for semantic description of the behaviors of individual resources. GPF defines as its part the Resource Goal/Behavior Description Framework (RgbDF). An implementation of GPF is supposed to include also appropriate RgbDF-based domain ontology, an appropriate RgbDF engine and a family of *semantic adapters to behaviors* to provide an opportunity to transform data from a variety of possible behavior representation standards and formats to RgbDF-based and back. See more on RgbDF in (Kaykova *et al.*, 2005c). The Semantic Agent Programming Language (S-APL), which is the core language of our platform, can be seen as a simplified realization of RgbDF.

The third is the *General Networking Framework (GNF)* for coordination and integration. GNF has to provide means for description of a group behavior within a business process. It specifies the Resource Process/Integration Description Framework (RpiDF), and an implementation of GNF is supposed to include also an appropriate RpiDF-based domain ontology, an appropriate RpiDF engine and a family of *semantic adapters to business process*es to provide opportunity to transform data from a variety of business process representation standards and formats to

RpiDF-based and back. The work on GNF is still ongoing in the UBIWARE project. Some of important GNF-related issues will be discussed in the next subsection. See more about contextual extension of RDF in (Khriyenko and Terziyan, 2006).The main ideas behind these three frameworks and the conceptual difference between RscDF, RgbDF and RpiDF are shown in Figure 2.

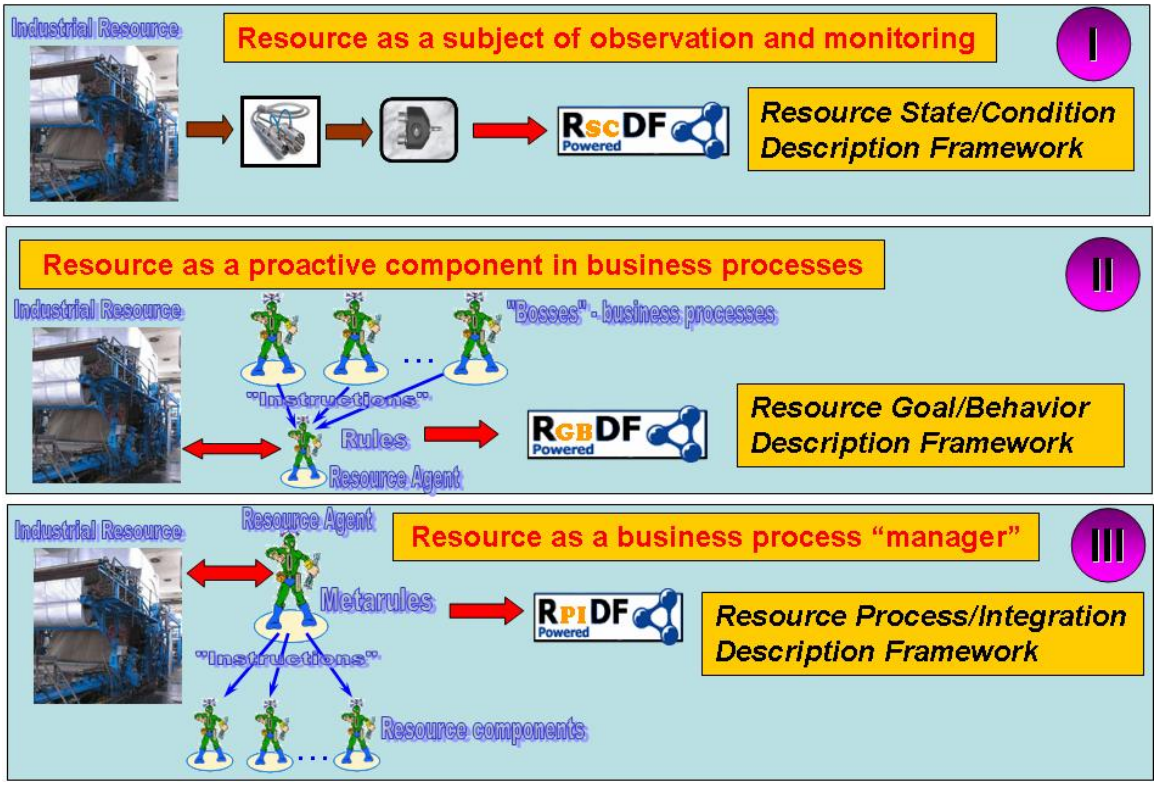

Fig. 2. The three frameworks in the Global Understanding Environment

As it was mentioned above, the GUN environment is supposed to have decision making resources, which utilize some machine learning algorithms. By getting data from some external industrial resources (devices, machines, etc.), such algorithms are to be able to build models for diagnostics and performance prediction of these devices. Natural heterogeneity and distribution of these algorithms and models result to another important challenge of GUN environment,

which is to provide an opportunity for use of automated algorithms (learning and decision making), and for models discovery, sharing, reuse, interoperability, invocation, integration and composition.

GUN Issues

Industrial World (as a natural part of the World of Things) consists of a variety of entities: simple and complex products, machines, tools, devices and their components, Web-services, human workers and customers, processes, software and information systems, standards, markets, domain ontologies, and others. Thus, the Industrial World contains all type of entities: physical, biological, and digital. On the other hand, the World of GUN also consists of a variety of entities: agents for managing Industrial World (IW) or GUN resources, resource histories semantically enriched with metadata, GUN ontologies, adapters for connecting with IW resources, tools, platforms, standards, executable software components, engines and rules employed by agents, multi-agent commitments and conventions, internal and global standards. GUN is meant for intelligent control over the Industrial World.

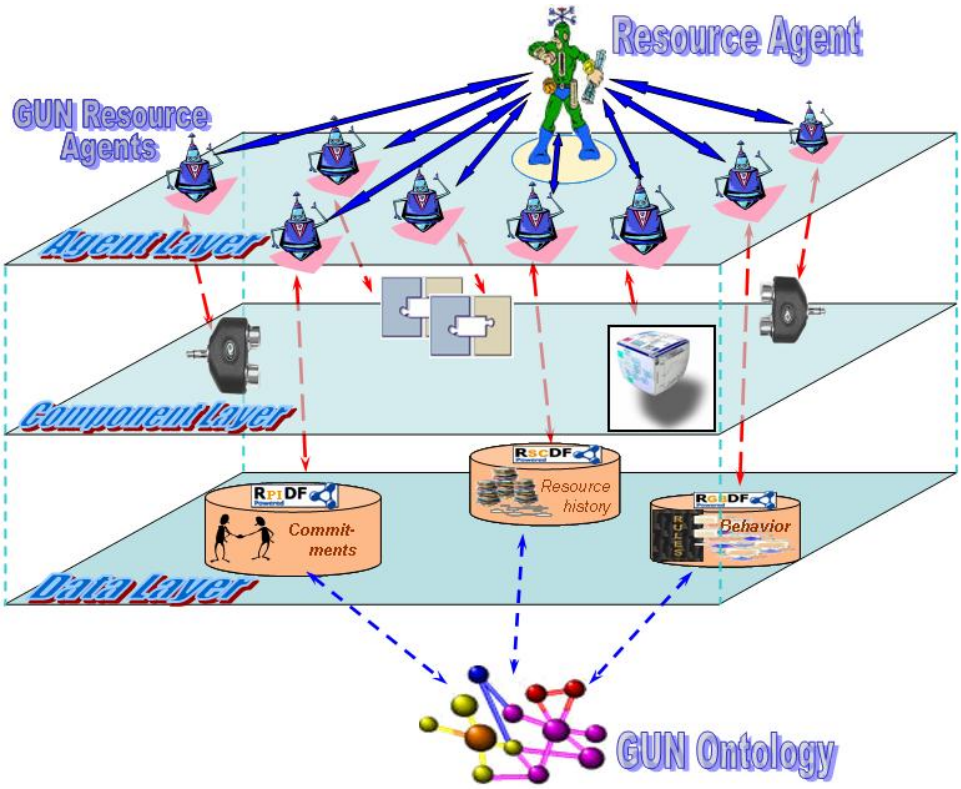

Fig.3. 3-layered GUN architecture for managing industrial resource

To each entity (resource) of the Industrial World, the Global Understanding Environment assigns a Resource Agent, which is assumed to "take care" of the resource and to implement GUN-level proactivity of the resource behavior. Thus, each of the IW resources can be GUN-supported if there is an opportunity to physically and semantically connect the resource to GUN. Heterogeneous IW resources, due to being represented by agents, become in a sense homogeneous in the GUN environment and naturally interoperable with other IW resources. Each GUN agent responsible for an industrial resource, *Resource Agent*, communicates with other agents, either with other resource agents or with *GUN resource agents*, and may even have no direct contact with any other software or other entities. Each GUN agent responsible for a GUN resource, *GUN Resource Agent,* necessarily communicates not only with other agents but also with the corresponding GUN resource directly.

In Figure 3, the 3-layered GUN architecture for a particular industrial resource management is shown (do not confuse with the architecture of a resource agent itself, which is 3-layered as well – see Section 4). The Agent Layer contains a resource agent who is responsible for a resource and also several GUN agents responsible for various software components needed for resource sensing, adaptation, condition monitoring, decision-making, maintenance, etc. Each GUN resource agent is connected to an appropriate software component on the Component Layer (e.g. resource sensor adapter, resource actuator adapter, alarm manager, etc.) and is able to invoke it whenever needed. Alternatively, it can be connected to an appropriate semantic storage on the Data Layer (automatically annotated resource history, resource proactive behavior, or resource commitments with other resources). Data Layer components are linked to the GUN ontology (either distributed or centralized), which contains necessary reusable patterns for resource history, resource behavior and resource coordination. Each resource agent keeps record of the resource states and own mental states in a semantically-rich RscDF-based format with link to the industrial domain ontology. Each resource agent keeps a set of needed rules and behavior patterns according to its role in a business process in a semantically-rich RgbDF-based format with link to GUN ontology. Each agent can keep needed adapters, histories, behavior sets, software components, commitments and reusable coordination patterns (in RpiDF-based format) and other GUN resources on own platform. Shared ontology guarantees interoperability and understanding among resource agents. Industrial world will be represented in the GUN environment with a distributed history database, which can be queried by agents and is the subject of agent communication. All the components of the Component Layer and the Data Layer can be exchanged between the resource agents, flexibly composed and reconfigured on-the-fly as result of context-driven agent coordination on the Agent Layer.

The history for a resource contains (see Figure 4) temporal tracks of semantically annotated: resource states as result of sensing and measurements; symptoms as detected by an embedded alarm system; diagnoses made by various experts or Web-services; maintenance (treatment) plans and activities made by experts or Web-services to fix recognized problems. Such a smart history may be not only subject to querying, sharing, integration, etc, but also can provide useful patterns (discovered by data-mining tools), which can be used for predictive diagnostics, maintenance, etc.

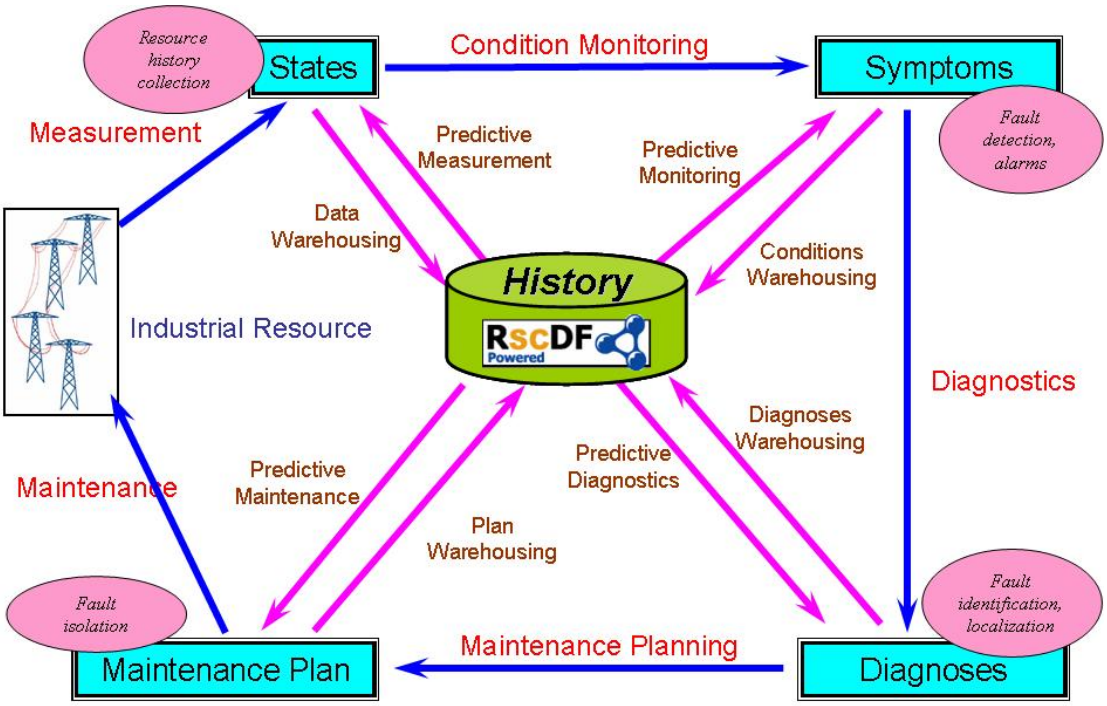

Fig. 4. Resource History and the process of its management

The General Networking Framework considers an opportunity of ontological modeling of business processes as integration of the behavioral models of various business actors (agents representing smart resources in the Web) in such a way that this integration will constitute the behavioral model of an agent responsible for the *alliance* of the components. This means that

such a corporate agent will monitor the behaviors of the proactive components against the constraints defined in the integration scenario. Such model is naturally recursive and this means that the corporate agent can be a component in a more complex business process and will be itself monitored by an agent from a more higher level of the hierarchy. The hierarchy of agents can be considered as possible mapping from the part-of ontological hierarchy of the domain resources (see Figure 5). Note that we do not assume centralized decision making, but only centralized supervision of compliance.

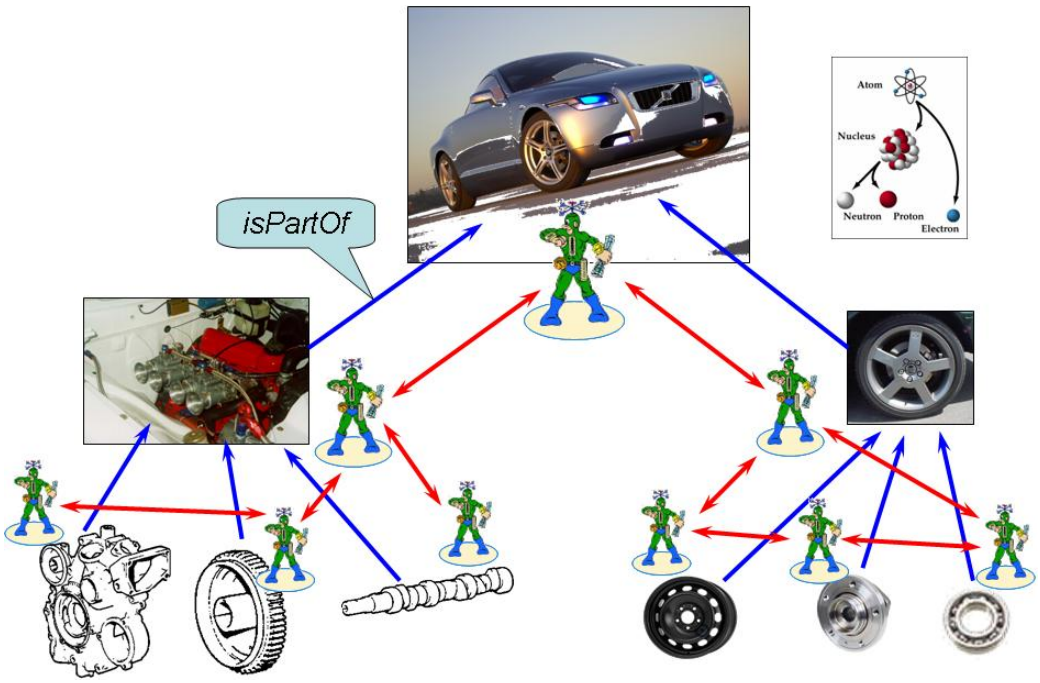

Fig. 5. Part-of hierarchy of resources results in corresponding hierarchy of agents

Another important concern is: What is a process in the GUN environment? Consider the following two axioms of GUN:

Axiom 1: Each resource in a dynamic Industrial World is a process and each process in this world is a resource.

Axiom 2: Hierarchy of subordination among resource agents in GUN corresponds to the *part-of* hierarchy of the Industrial World resources.

As all GUN resources, a process has own properties that describe process's state, history, sub processes and belongingness to upper-process (super-process). Thus, following the principles of GUN resource, each process should be assigned an agent that serves this process, similarly to any other resource. GUN's Top Agent is the one, which resource to be taken care of is the Industrial World as whole. Such agent will be on the top of the hierarchy of resource agents.

Each industrial resource can theoretically be involved into several processes, appropriate commitments and activities, which can be either supplementary or contradictory. This means that the resource is part of several more complex resources and its role within each is probably different. Modeling such resources with GUN can be done with the resource agent that can make clones of itself and distribute all necessary roles among them (see Figure 6).

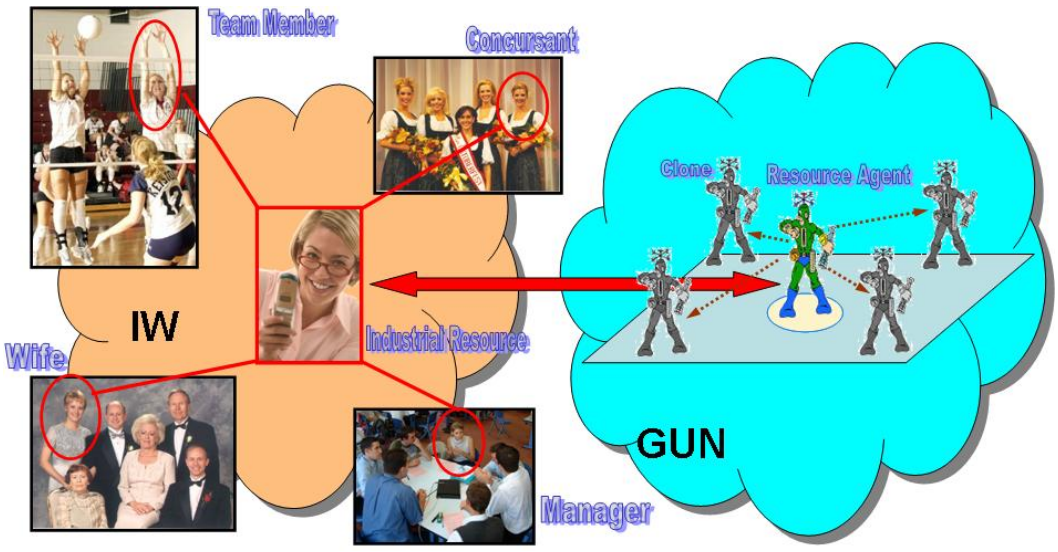

Fig. 6. Multiple roles of a resource in the Industrial World and appropriate agent-clones in GUN

Each industrial resource, which joins some commitment, will behave according to the restrictions that the rules of that commitment imply. The more commitments individual resource takes, the more restriction will be put on its behavior (see Figure 7).

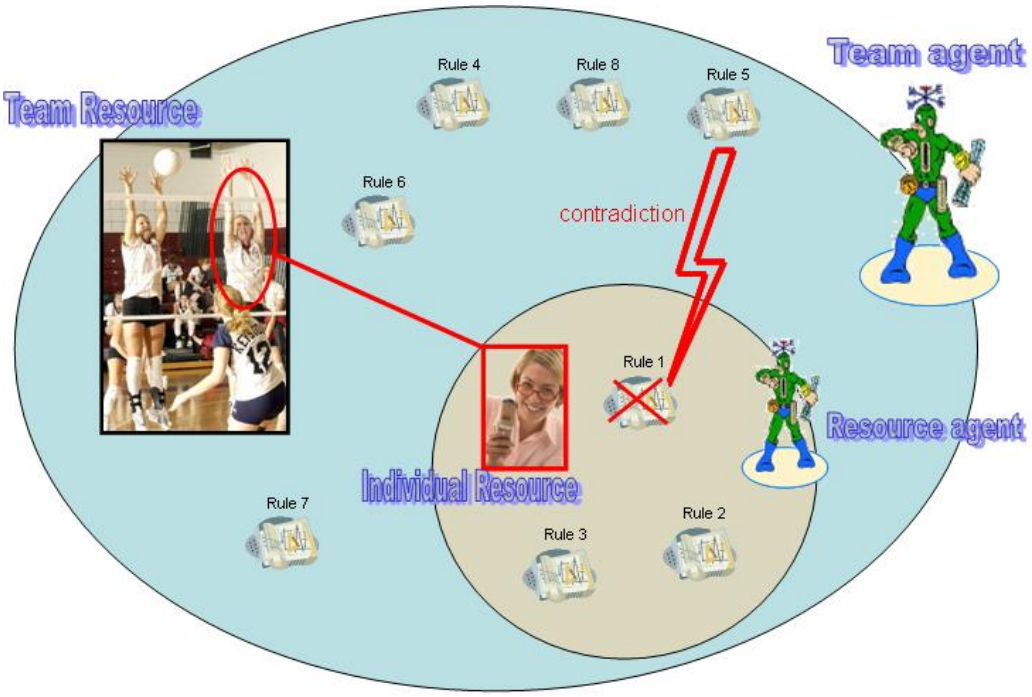

Fig. 7. Individual vs. team resource freedom

An important feature of the General Networking Framework is the smart way of managing commitments (processes and contracts) of proactive resources to enable cooperation of them and to reach the group's goals along with the individual ones.

Summarizing we can say that GUN vision assumes proactivity of all the heterogeneous resources (humans, devices, services) in the World of Things and intends to provide reusable behaviors and reusable coordination patterns to all the resources, making them components in a self-organized process of automatic creation of complex dynamic reconfigurable systems for different industrial purposes. GUN vision allows considering everything as a smart agent-driven

resource, which are not only physical objects, but also processes, mathematical models, ontologies and even messages in communication. The last one be used to allow if needed dynamic smart routing, where a smart message itself (not the nodes) decides where to go further within a network, is able to collect own history and communicate with others.

In the following section, we provide more details on the implementation issues focusing on integration of MAS platforms with semantic representation of reusable behavioral patterns for the resource agents.

THE CURRENT STATE OF THE REALIZATION

This section describes the currently achieved state of the GUN realization in the UBIWARE platform. The central to the platform is the architecture of a SmartResource agent depicted in Figure 8.

The basic 3-layer agent structure is similar to, for example, the Agent Factory's ALPHA/AFAPL (see Section 2). There is the behavior engine implemented in Java, a declarative middle layer, and a set of sensors and actuators which are again Java components. The latter we refer to as *Reusable Atomic Behaviors (RABs)*. We do not restrict RABs to be only sensors or actuators, i.e. components sensing or affecting the agent's environment. A RAB can also be a reasoner (data processor) if some of the logic needed is impossible or is not efficient to realize with the S-APL means, or if one wants to enable an agent to do some other kind of reasoning beyond the rule-based one.

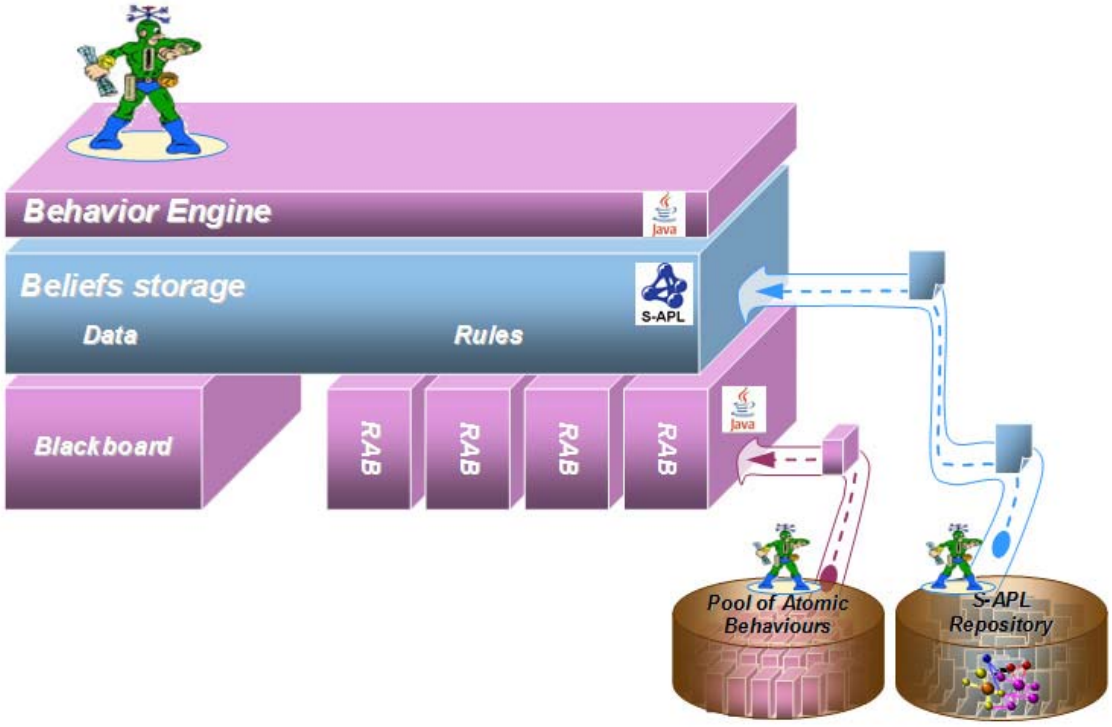

Fig. 8. The agent architecture

The middle layer is the S-APL beliefs storage. As explained in Section 2, the main factor that differentiates S-APL from traditional APLs like AgentSpeak or ALPHA is that S-APL is RDF-based. An additional immediate advantage is that in S-APL the difference between the data and the program code (rules and plans) is only logical but not any principal. They use the same storage, not two separate. This also means that: a rule upon its execution can add or remove another rule, the existence or absence of a rule can be used as a premise of another rule, and so on. None of these is normally possible in traditional APLs treating rules as special data structures principally different from normal beliefs which is n-ary predicates. S-APL is a very symmetric in this respect – anything that can be done to or with a simple statement can also be done to any structure of any complexity.

Technically, our implementation is built on the top of the Java Agent Development Framework (JADE, Bellifemine et al. 2007), which is a Java implementation of IEEE FIPA specifications. The S-APL behavior engine is an extension (subclass) of JADE's Agent class, while the base class for all RABs is an extension of JADE's SimpleBehavior class.

As Figure 8 stresses, an S-APL agent can obtain the needed data and rules not only from local or network documents, but also through querying S-APL repositories. Such a repository, for example, can be maintained by some organization and include prescriptions (lists of duties) corresponding to the organizational roles that the agents are supposed to play. 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 in Section 2, the agents not enacting a particular role can still make some use of 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.

In our implementation, such querying is performed as inter-agent action with FIPA ACL messaging, but does not involve any query or content languages beyond S-APL itself.

As can be seen from Figure 8, agents also can to load RABs remotely. This is done as an exception mechanism triggered when a rule prescribes engaging a RAB while the agent does not have it available. Thus, organizations are able to provide not only the rules to follow but also the tools needed for 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.

We also equip each agent with a blackboard, through which RABs can exchange arbitrary Java objects. Similar solution can be found e.g. in the Cougaar framework (Helsinger, 2004). The reason for that is not to unnecessarily restrict the range of applications that could be realized with S-APL. Without such a blackboard, RABs would be always forced to translate all data into RDF (even when the S-APL code of the agent is not concerned with the content of data, or could not process it) or at least serialize it as text string to put the as object of a statement. This could restrict the performance and, more importantly, significantly reduce the wish to use S-APL. Blackboard is also necessary to accommodate objects like Socket, HttpServletResponse or similar to enable an agent to process and respond to HTTP requests, which may be needed in many applications. With the blackboard extension, the developers of a specific application can use S-APL in different ways:

- Semantic Reasoning. S-APL rules operating on S-APL data.
- Semantic Data. RABs (i.e. Java components) operating on S-APL semantic data.
- Workflow management. RABs operating on Java blackboard objects, with S-APL used only as workflow management tool, specifying what RABs are engaged and when.

- Any combination of the three options above.

A CASE: DISTRIBUTED POWER NETWORK MAINTENANCE

This section describes a case study in the domain of distributed power network maintenance we have been performing, starting from early 2006, in collaboration with ABB (Distribution Automation). The goal is to study the potential add-value which ABB could receive from introducing Semantic Web technologies and 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 first subsection 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. The second subsection reports then on the developed prototype, which also demonstrates some of the basic features of the UBIWARE platform described in Section 4.

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, which belong to the MicroSCADA Pro product family, like DMS 600 or MicroSCADA Pro Distribution Management System and SYS 600, which is MicroSCADA Pro Control System. These systems provide an integrated graphical view over the sub-network, provide 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.

Fig. 9. Scenario: sub-networks' interoperability

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 GUN system could solve this issue (Figure 9). The information flow will go through the agents representing the sub-networks in the GUN environment. 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 10). ABB will be always able to update algorithms, add new, and so on. The GUN environment 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 GUN. 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.

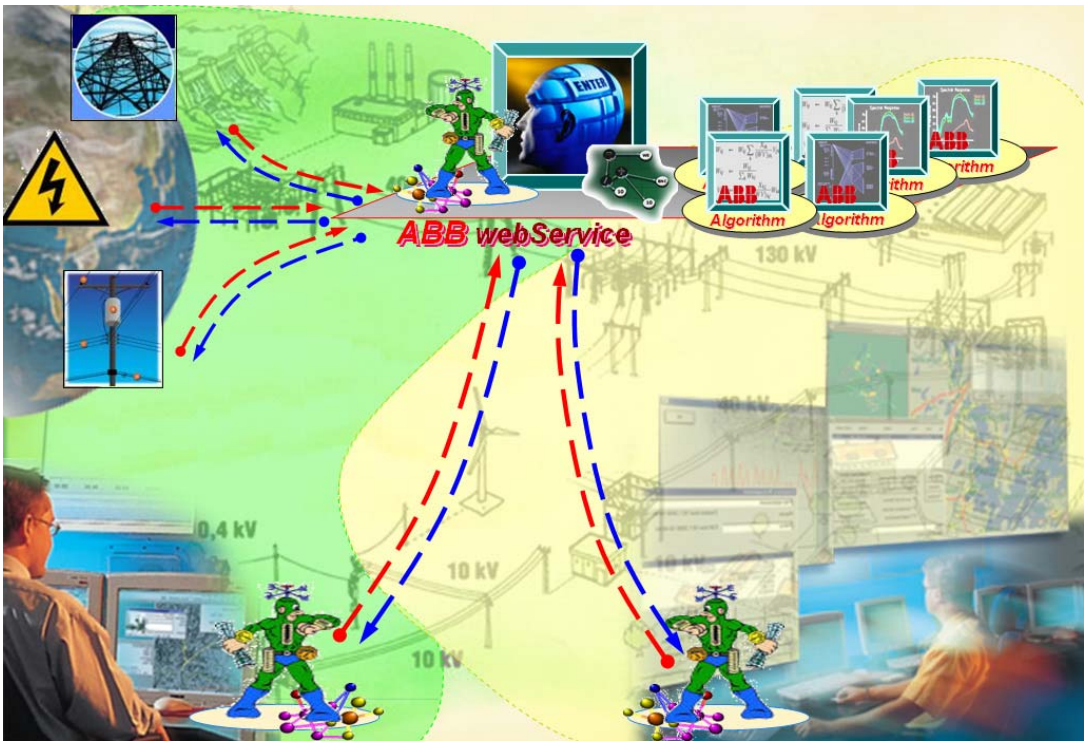

Fig. 10. Scenario: a new business model

Fig. 11. Scenario: integration with external information services

The third scenario in our vision is related to the possibility of 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 11). 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 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 DMS); also, dynamically-changing information can be accessed and represented on the interface.

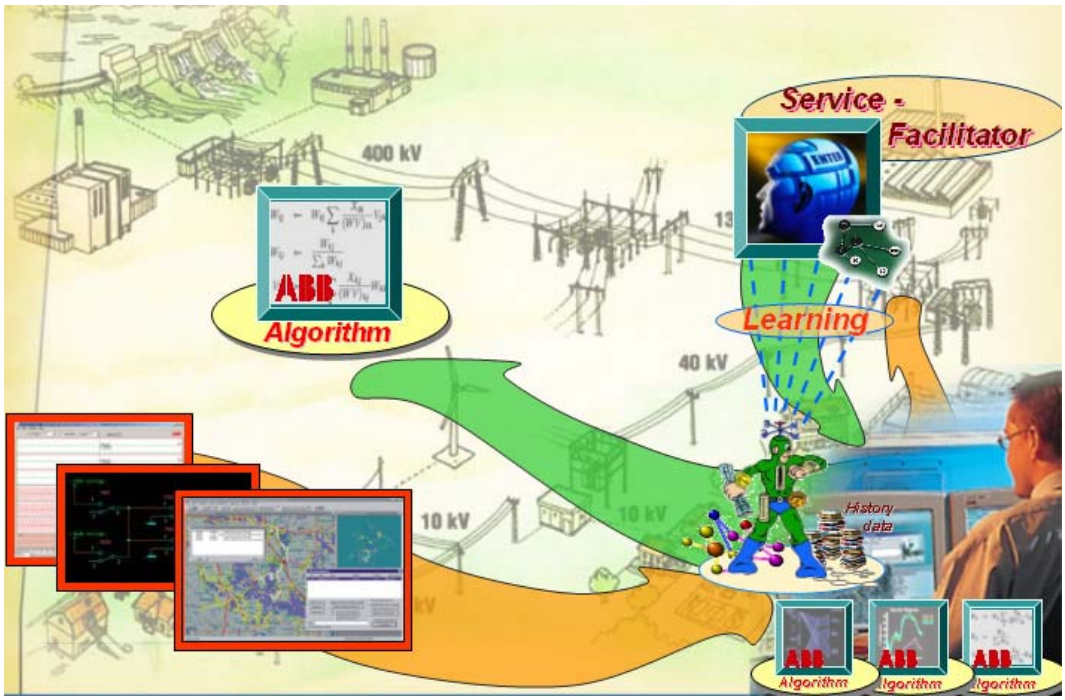

Fig. 12. Scenario: expert's knowledge transfer

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 12). 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 existing 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.

The Prototype

We also developed a prototype, mainly for the purpose of the concept demonstration, both for ABB and their customers. The 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 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 10 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 in GUN environment by a corresponding agent (the service itself is a stub in the prototype).

- *Weather Service*. A service providing constantly updated 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 4.

In the 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 13 shows the process of starting up an agent. The same process is followed for every new agent on the UBIWARE platform. From the startup batch file of the 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 13, 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 “Feeder1” – 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. Additionally, it is supposed to have a capability “RABLoader” - needed for remote loading of RABs. First, the agent “Feeder1” loads the startup.sapl 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 three scripts 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.

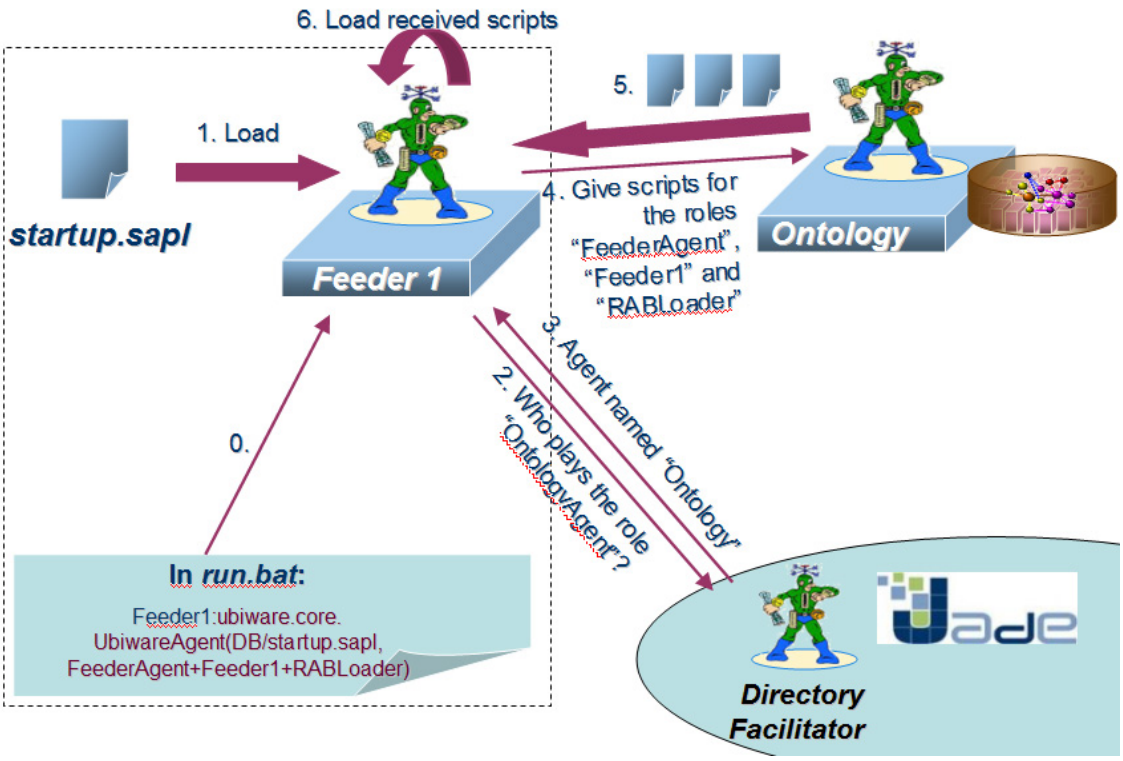

Fig. 13. An agent's start-up

Figure 14 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.

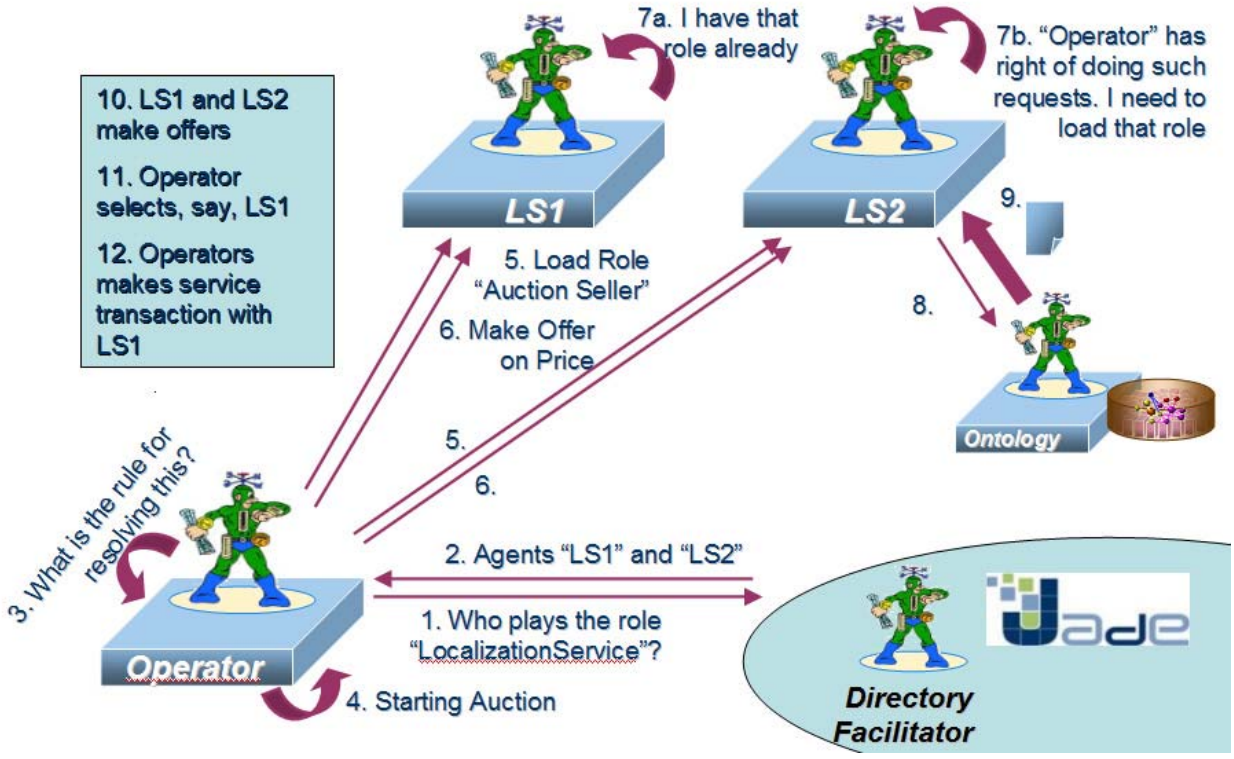

Fig. 14. Auction for selection of the service provider

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.

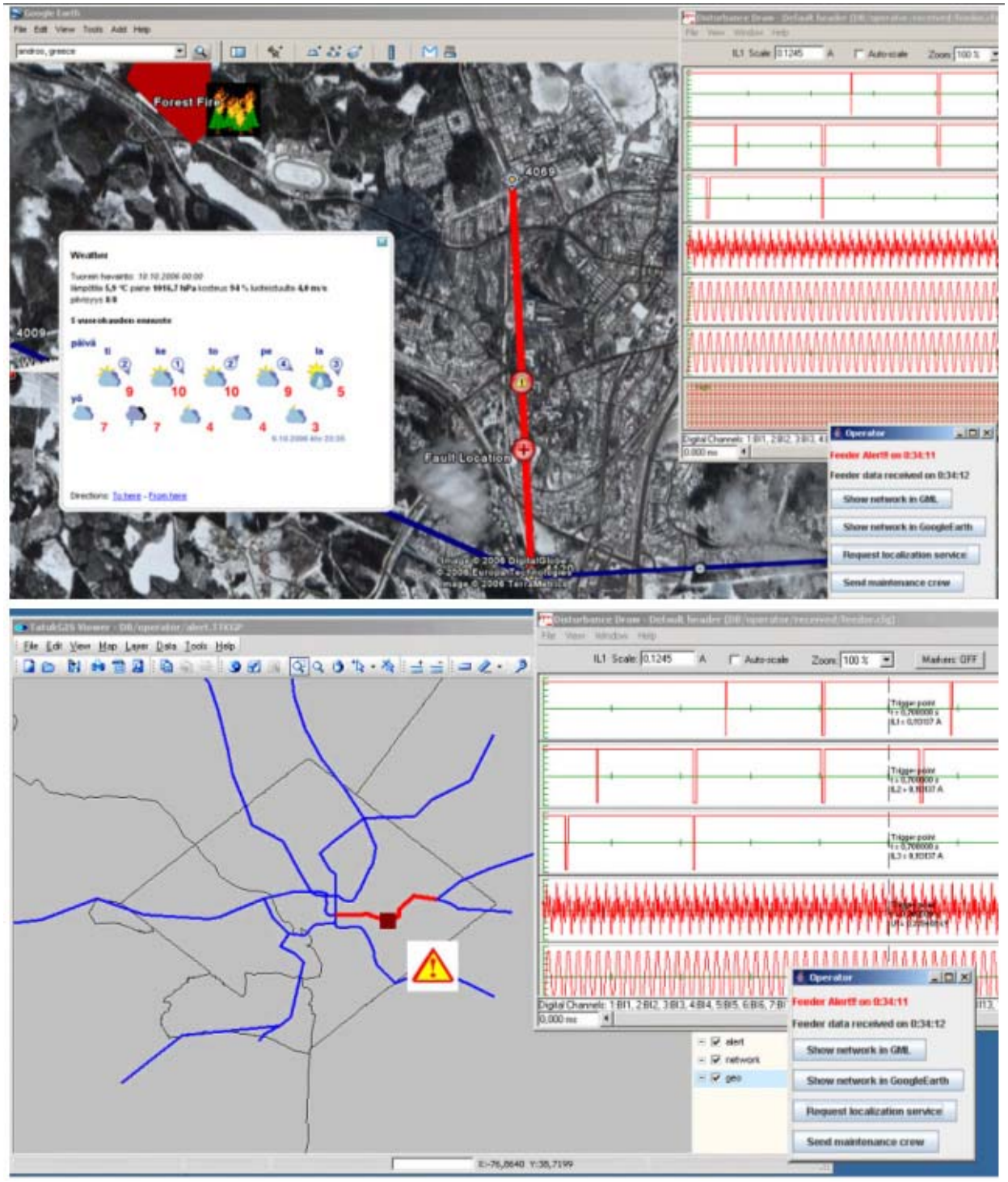

Fig. 15. Interface of an operator provided by his/her agent (2 versions)

Figure 15 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.

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 real-time data on the 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.

DISCUSSION AND FUTURE WORK

In this chapter, we described our work on the Global Understanding Environment, a general middleware framework aiming at providing means for building complex industrial systems consisting of components of different nature, based on the Semantic Web and agent technologies. From the Semantic Web point of view, GUN could probably be referred to as

Proactive Self-Managed Semantic Web of Things. We believe that such Proactive Self-Managed Semantic Web of Things can be the future killer application for the Semantic Web.

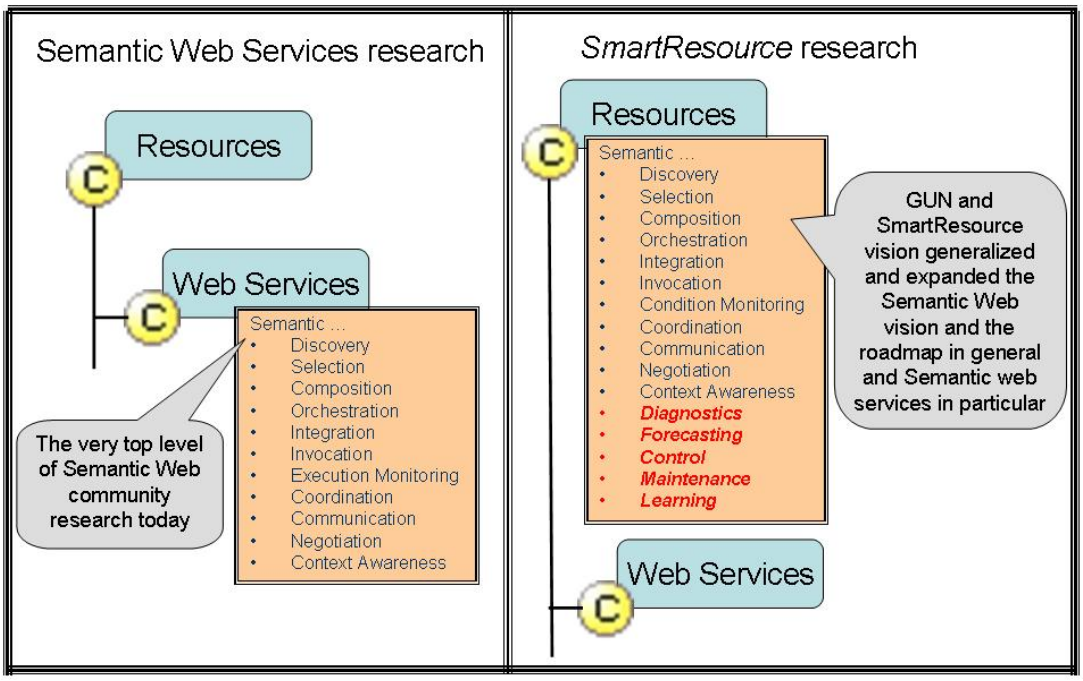

Fig. 16. Shifting Semantic Web roadmap to the Web of Things domain

The shift from the Web of documents and software to the Internet of Things should affect the Semantic Web research roadmap. So far, the concepts of (semantic) discovery, selection, composition, orchestration, integration, invocation, execution monitoring, coordination, communication, negotiation, context awareness, etc. were mainly applied to the Web-services domain. In future, however, all these concepts should be modified to be applicable also to a resource from the Internet of Things. Also, some new things should be taken into account, such as e.g. (semantic) diagnostics, forecasting, control, maintenance, learning, etc. (see Figure 16).

Our research on GUN in the SmartResource project (2004-2006) has pointed out the need of updating RDF as the basic Semantic Web framework – in three dimensions: regarding context-sensitivity and dynamics, proactivity, and coordination (see Figure 17).

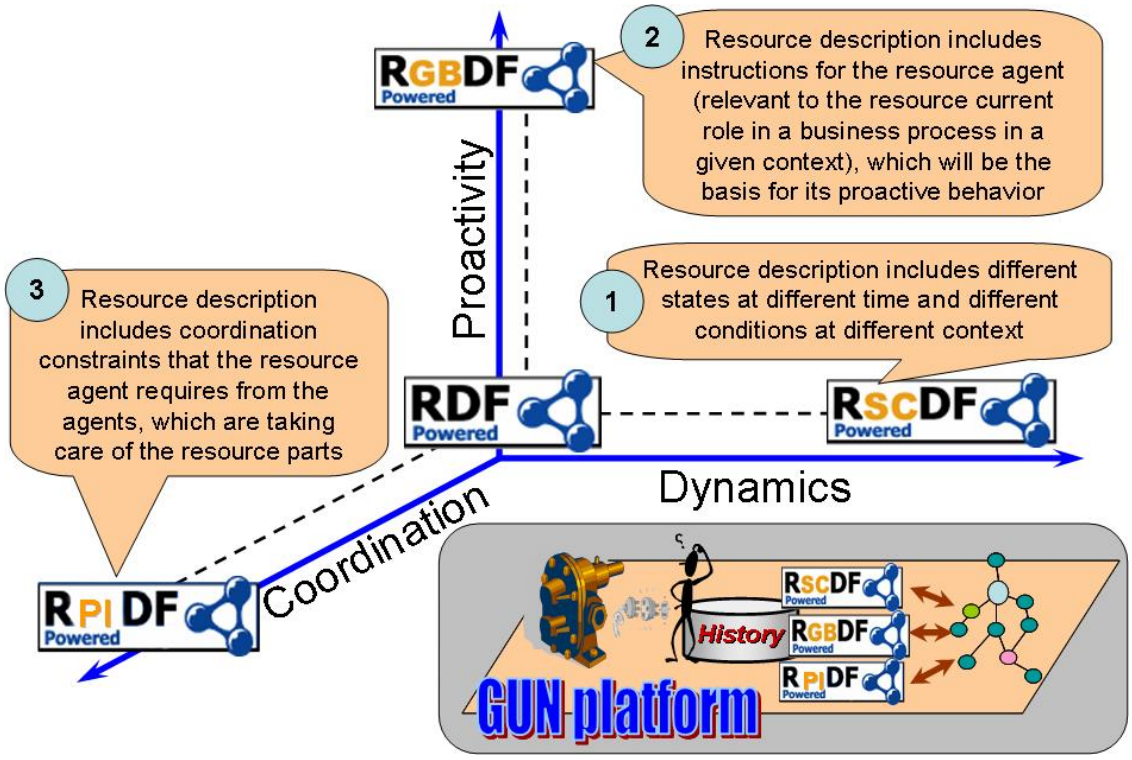

Fig. 17. Three dimensions of developing RDF towards the Web-of Things domain

Our plans include further development the UBIWARE platform towards of a general domain-independent tool allowing creation of self-managed complex industrial systems consisting of mobile, distributed, heterogeneous, shared and reusable components. Those components can be smart machines and devices, sensors, actuators, RFIDs, web-services, software, information systems, communication networks, humans, models, processes, organizations, etc. Such middleware will enable various components to automatically discover each other and to configure a system with complex functionality based on the atomic functionalities of the components.

In this work, we will naturally integrate the Ubiquitous Computing domain with such domains as Semantic Web, Proactive Computing, Autonomous Computing, Human-Centric Computing, Distributed AI, Service-Oriented Architecture, Security and Privacy, and Enterprise Application Integration. UBIWARE aims at bringing the following values to the industrial automation domain: Openness, Intelligence, Dynamics, Self-Organization, Seamless Services and Interconnectivity, Flexibility and Re-configurability, Context-Awareness, Semantics, Proactivity, Interoperability, Adaptation and Personalization, Integration, Automation, Security, Privacy and Trust.

Utilization of the Semantic Web technology in UBIWARE enables:

- Reusable configuration patterns for ubiquitous resource adapters;
- Reusable semantic history blogs for all ubiquitous components;
- Reusable semantic behavior patterns for agents and processes descriptions;
- Reusable coordination, integration, composition and configuration patterns;
- Reusable decision-making patterns;
- Reusable interface patterns;
- Reusable security and privacy policies;

In this chapter, we had focused, among other things, on proactive agent-driven functionality of industrial resources supported by the UBIWARE platform. The existing architecture allows an agent to decide which role it should take in changing context, download and apply reusable

semantic description of behavior appropriate to the role. Many interesting issues remain with respect to this for further study, for example:

- Possible conflicts between the roles simultaneously played by the same agent, i.e. what in this case should be the self-coordination mechanism;
- Principles and concrete benefit of using the code describing a behavior model of one agent by other agents.

ACKNOWLEDGEMENTS

We are grateful to Tekes (Finnish National Agency for Technology and Innovation), Agora Center of the University of Jyväskylä, and cooperating companies (ABB, Metso Automation, Fingrid, TeliaSonera, TietoEnator, Inno-W, and Jyväskylä Science Park) for supporting activities of the SmartResource and UBIWARE projects. Special thanks to ABB for collaboration on the case study reported in this paper, and permission to publish the results. Thanks to Oleksiy Khriyenko for creating some of the figures for this paper.

REFERENCES

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). Developing Multi-Agent Systems with JADE. Wiley

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. Scientific American, 284(5), 34-43.

Bosse, T., and Treur, J. (2006). Formal interpretation of a multi-agent society as a single agent. Journal of Artificial Societies and Social Simulation 9(2).

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203-236.

Brock, D.L. and Schuster, E. W. (2006). On the semantic web of things, Semantic Days 2006, Norway, April 26, 2006.

Buckley, J. (2006). From RFID to the Internet of Things: Pervasive Networked Systems, Final Report on the Conference organized by DG Information Society and Media, Networks and Communication Technologies Directorate, CCAB, Brussels (available in: http://www.rfidconsultation.eu/docs/ficheiros/WS_1_Final_report_27_Mar.pdf).

Collier, R., Ross, R. and O'Hare, G.M.P. (2005). Realising reusable agent behaviours with ALPHA. In Proc. 3rd Conference on Multi-Agent System Technologies (MATES-05), LNCS vol. 3550, pp. 210–215.

Dastani, M., van Riemsdijk, B., Dignum, F., and Meyer, J.-J.Ch. (2003). A programming language for cognitive agents: Goal directed 3APL. Proc. First International Workshop on Programming Multi-Agent Systems, LNCS vol. 3067, pp. 111-130.

Freeman, P. A. (2006). Statement before the Committee on Science of the U.S. House of Representatives, Hearing on Innovation and Information Technology: The Government, University and Industry Roles in Information Technology Research and Commercialization, Austin, Texas (available in: http://www.house.gov/science/hearings/full06/May%205/ Freeman.pdf).

Helsinger, A., Thome, M., and Wright, T. (2004). Cougaar: a scalable, distributed multi-agent architecture. In Proc. IEEE International Conference on Systems, Man and Cybernetics. Volume 2, pp. 1910–1917.

Jennings, N.R., Sycara K. P., and Wooldridge, M. (1998). A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems 1(1), 7-38.

Jennings, N.R. (2000). On agent-based software engineering. Artificial Intelligence 117(2), 277-296.

Jennings, N.R. (2001). An agent-based approach for building complex software systems. Communications of the ACM 44(4), 35-41.

Katasonov, A. (2008) UBIWARE Platform and Semantic Agent Programming Language (S-APL): Developer's guide, Online: http://users.jyu.fi/~akataso/SAPLguide.pdf.

Katasonov, A. and Terziyan, V. (2007). SmartResource Platform and Semantic Agent Programming Language (S-APL). In Proc. 5th Conf. Multi-Agent Technologies (MATES'07), LNAI vol. 4687, pp. 25-36.

Kaykova O., Khriyenko O., Kovtun D., Naumenko A., Terziyan V., and Zharko A. (2005a). General Adaption Framework: Enabling Interoperability for Industrial Web Resources. International Journal on Semantic Web and Information Systems, 1(3), 31-63, Idea Group.

Kaykova O., Khriyenko O., Naumenko A., Terziyan V., and Zharko A. (2005b). RSCDF: A Dynamic and Context Sensitive Metadata Description Framework for Industrial Resources. Eastern-European Journal of Enterprise Technologies 3(2), 55-78.

Kaykova O., Khriyenko O., Terziyan V., and Zharko A. (2005c). RGBDF: Resource Goal and Behaviour Description Framework. In Proc. 1st International Conference on Industrial Applications of Semantic Web, Springer, IFIP, vol.188, pp. 83-99.

Khriyenko O., and Terziyan V. (2006). A Framework for Context-Sensitive Metadata Description. International Journal of Metadata, Semantics and Ontologies 1(2), 154-164.

Kephart J. O. and Chess D. M. (2003). The vision of autonomic computing, IEEE Computer 36(1), 41-50.

Krummenacher, R., and Strang, T. (2005). Ubiquitous Semantic Spaces, In Conference Supplement to the 7th Intl. Conf on Ubiquitous Computing (UbiComp 2005), Tokyo.

Lassila, O. (2005a) Applying Semantic Web in Mobile and Ubiquitous Computing: Will Policy-Awareness Help?, in Proc. Semantic Web Policy Workshop, 4th International Semantic Web Conference, Galway, Ireland, pp. 6-11.

Lassila, O. (2005b). Using the Semantic Web in Mobile and Ubiquitous Computing, In Proc. 1st IFIP Conference on Industrial Applications of Semantic Web, Springer IFIP, pp. 19-25.

Lassila, O., and Adler, M. (2003). Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web, In: D. Fensel et al. (eds.), Spinning the Semantic Web, MIT Press, pp. 363-376.

Motta, E., and Sabou, M. (2006). Next Generation Semantic Web Applications, In Proc. 1st Asian Semantic Web Conference (ASWC), Beijing, China.

Qasem, A., Heflin J., and Mucoz-Avila H. (2004). Efficient Source Discovery and Service Composition for Ubiquitous Computing Environments. In: Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications, ISWC 2004.

Rao, A.S. and Georgeff, M.P. (1991). Modeling rational agents within a BDI architecture. Proc. 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91), pp. 473-484.

Rao, A.S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. Proc. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNCS vol.1038, pp. 42-55.

Shoham, Y. (1993) Agent-oriented programming. Artificial Intelligence, 60(1), 51–92.

Sabou, M., Lopez, V., and Motta, E. (2006). Ontology Selection on the Real Semantic Web: How to Cover the Queens Birthday Dinner?, In Proceedings of EKAW, Podebrady, Czech Republic.

Tamma, V.A.M., Aart, C., Moyaux, T., Paurobally, S., Lithgow-Smith, B., and Wooldridge, M. (2005). An ontological framework for dynamic coordination. Proc. 4th International Semantic Web Conference'05, LNCS vol. 3729, pp. 638-652.

Terziyan V. (2003) Semantic Web Services for Smart Devices in a "Global Understanding Environment", In: On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, LNCS vol. 2889, Springer-Verlag, pp.279-291.

Terziyan V. (2005). Semantic Web Services for Smart Devices Based on Mobile Agents, International Journal of Intelligent Information Technologies, 1(2), 43-55, Idea Group.

Vázquez-Salceda, J., Dignum, V., and Dignum, F. (2005). Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems 11(3), 307-360.

Wooldridge, M. (1997). Agent-based software engineering. IEE Proceedings of Software Engineering 144(1), 26-37.

Wooldridge, M., Jennings, N.R., and Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3), 285-312.

Further Reading:

1. Schuster, E.W., Allen, S.J. and Brock, D.L. *Global RFID*, Springer, October 2006
2. Knowledge Media Institute publications on the concept of "Real Semantic Web", available in: http://kmi.open.ac.uk/publications/publications.cfm?id=110.
3. Aware.IT publications on the synergies of pervasive computing and the Semantic Web, available in: http://www.awareit.com/blog/index.php?/pages/publications.html
4. Publications of the TAPAS Project (Telematics Architecture for Play-Based Adaptable System), available in http://tapas.item.ntnu.no/wiki/index.php/Publications

Useful URLs:

1. Public web-site of the SmartResource project: http://www.cs.jyu.fi/ai/OntoGroup/SmartResource_details.htm
2. Public web-site of the UBIWARE project: http://www.cs.jyu.fi/ai/OntoGroup/UBIWARE_details.htm
3. IBM Research - Autonomic Computing: http://researchweb.watson.ibm.com/autonomic/
4. Semantic Web on The World Wide Web Consortium (W3C): http://www.w3.org/2001/sw/
5. IEEE Foundation for Intelligent Physical Agents (FIPA): http://www.fipa.org/

Possible Papers Titles/ Essays:

1. *Industrial Automation and Semantics*: the specifics of the industrial automation domain from the point of view of how the semantic technology can be applied there.
2. *Agents for the Semantic Web or Semantic Technologies for the Agents*: the added value that the semantic technologies can provide for multi-agent systems.
3. *Realizing the Semantic Web of Things*: the factors, barriers and time needed to implement it.
4. *Organizing Industrial Multi-Agent Systems*: advantages and disadvantages of hierarchical, decentralized and other organizational models in the context of industrial automation applications.