Semantic descriptions of resources with proactive behavior of autonomous condition monitoring applications

Gunjan Mishra, Diwakar Yagyasen

Abstract— The goal of the semantic web is to be "a web talking to machines", i.e. in which machines can provide a better help to people because they can take advantage of the content of the Web. The information on the web should thus be expressed in a meaningful way accessible to computers. Semantic Web aims to improve upon the meaning, in machine-understandable terms, of information currently available on the world-wide-web. This enables computers, in the form of autonomous software agents, to work with the wealth of world-wide-web information more easily. Moreover, it enhances the human-computer co-operation by bringing the concept of human understanding closer to the machine. Autonomous systems must be automatic and, in addition, they must have a capacity to form and adapt their behaviour while operating in the environment. Thus traditional AI systems and most robots are automatic but not autonomous - they are not fully independent from the control provided by their designers. Autonomous systems are independent and are able to perform self-control.

Index Terms— Sementic web,web ontology language,sementic agent programming language,resourse description framework,sementic web middleware,agent communication in S-APL,proactive future internet

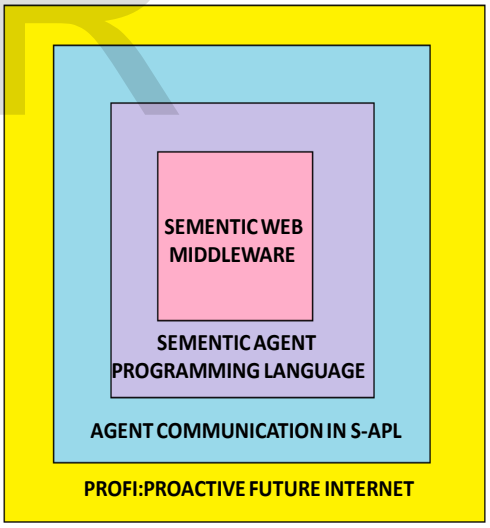
1 INTRODUCTION

The contribution of the ongoing Smart Resource project (2004 to 2006), together with appropriate research effort, includes prototype implementation of distributed Semantic Web enabled maintenance management environment with complex interactions of components, which are devices, humans (experts, operators), and remote diagnostic Web services. The environment will provide automatic discovery, integration, condition monitoring, remote diagnostics, and cooperative and learning capabilities of the heterogeneous resources to deal with maintenance problems. Maintenance (software) agents will be added to industrial devices, which are assumed to be interconnected in a decentralized peer-to-peer network and which can integrate diagnostic services in order to increase the maintenance performance for each individual device. The maintenance case is expected to demonstrate the benefits and possibilities of a new resource management framework and Semantic Web technology in general. An approach to that case harnesses the potential of emerging progressive technologies, such as Semantic Web, agent technology, machine learning, Web services, and peer-to-peer.

2 PROPOSED MODEL

The common goal of the middleware development initiatives is to develop a framework which provides a platform to semantic resources with proactive behaviour in autonomous condition monitoring applications. The middleware describe that the semantic web has a large capability to establish a connection between software agent and the internet. The middleware(fig.1) is like a platform to provide semantic web recourses easily handle by the user. In this paper the middleware is divided into three main component which is describe below-

- Semantic Web Middleware
- Semantic Agent Programming Language
- Agent communication in S-APL
- PROFI: Proactive Future Internet



Functional Components of Sementic Web Middleware

Figure 1:functional components of sementic web middleware

2.1 Semantic Web Middleware

Semantic web middleware consist of resource description framework (RDF) and web ontology language (OWL) which provides a better enabling platform for the

semantic resources treated as proactive. The semantic web middleware regarding to role of RDF and OWL is describe below:

2.1.1 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is an XML-based language for describing information contained in a Web resource. A resource can be a Web page, an entire Web site, or any item on the Web that contains information in some form. RDF enables the encoding, exchange, and reuse of structured metadata. It allows for metadata interoperability through the design of mechanisms that support common conventions of semantics, syntax, and structure. RDF makes no assumption about a particular application domain, nor defines the semantics of any particular application domain. The definition of the mechanism is domain neutral, yet the mechanism is suitable for describing information about any domain. RDF can be used in a variety of application areas including:

- Resource Discovery - RDF will enable search engines to more easily discover resources on the Web.
- Cataloging - RDF will enable users to better describe the content and content relationships available at a particular Web site, page, or digital library.
- Intelligent Software Agents -RDF will facilitate knowledge sharing and exchange, and allow software agents to more intelligently find, filter and merge data.
- Content Rating - RDF will allow content to be rated.
- Intellectual Property Rights - RDF will allow users to more easily express and enforce intellectual property rights of Web sites.
- Privacy Preferences and Privacy Policies - RDF will allow users and Web sites to express privacy preferences and site-wide privacy policies that can be interpreted by applications.
- Digital Signatures - RDF will be a key to building the "Web of Trust" for e-commerce, collaboration, and other applications.

2.1.2 Web Ontology Language (OWL):

OWL is an ontology language for the Web. It became a World Wide Web Consortium (W3C) Recommendation1 in February 2004. As such, it was designed to be compatible with the eXtensible Markup Language (XML) as well as other W3C standards. In particular, OWL extends the Resource Description Framework (RDF) and RDF Schema, two early Semantic Web standards endorsed by the W3C. Syntactically, an OWL ontology is a valid RDF document and as such also a well-formed XML document. This allows OWL to be processed by the wide range of XML and RDF tools already available.

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users.

- **OWL Lite**
  Supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL, see the section on OWL Lite in the OWL Reference for further details.
- **OWL DL**
  Supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied logics that form the formal foundation of OWL.
- **OWL Full**
  It is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

2.2 Semantic Agent Programming Language:

S-APL has as an axiom that anything inside an agent's mind is a belief [14]. All other mental attitudes such as goals, commitments, behavioral rules are just compound beliefs. Thus, an S-APL document is basically a statement of some agent's current or expected (by an organization) beliefs. S-APL is based on Notation3 (N3)[15] and utilizes the syntax for rules very similar to that of N3Logic[16]. N3 was proposed as a more compact, better readable and more expressive alternative to the dominant notation for RDF, which is RDF/XML. One special feature of N3 is the concept of formula that allows 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. In S-APL, we refer to formulae as context containers. The top level of the S-APL document, i.e. of what is the general truth for the agent, we refer to as general context or just G.

Below, we describe the main constructs of S-APL. We use three namespaces:"sapl:" for S-APL constructs, "java:" for RABs, and "p:" for RAB parameters. The empty namespace":" is used for resources that are assumed to be defined elsewhere. The two constructs below are equivalent and define a simple belief. The latter is introduced for syntactic reasons.

:room1 :hasTemperature 25
{:room1 :hasTemperature 25} sapl:is sapl:true

The next two constructs add context information:

{:room1 :hasTemperature 25} :measuredBy :sensor1
{:room1 :hasTemperature 25} sapl:is sapl:true ;
:measuredBy :sensor1

The former states that"sensor1 measured the temperature to be 25" without stating that "the agent believes that the temperature is 25". In contrast, the latter states both. This demonstrates a specific convention of S-APL: rather than doing several statements about one container,"{...} P O; P O" leads to linking the statements inside the formula to two different containers. Then, using sapl:true it is also possible to link some statements to a container and to one of its
nested containers.

The goals of the agent and the things that the agent believes to be false are defined, correspondingly, as:

sapl:I sapl:want {:room1 :hasTemperature 25}
{:room1 :hasTemperature 25} sapl:is sapl:false

2.3 Agent communication in S-APL

IEEE FIPA developed a set of standard specifications for agent communication including ACL for message envelopes and SL for contents [17]. While the standard position of ACL is unquestionable, the value of SL is less certain. Although meaning"semantic language", SL is not based on W3C's RDF semantic data model. Rather, SL follows the traditional agent design approaches where the agents' beliefs and thus also the atoms of their communications are nary predicates. However, N-ary predicates do not make the meaning of data as explicit as RDF triples do. Also, only the whole message can be linked to an ontology, as compared to the ability of RDF to link every individual resource to its own ontology, if needed.

In this section, we describe how we use S-APL as the content language in agent communications. Since one of the important communicative actions is querying for information, this role of S-APL overlaps with that of SPARQL. The problem with SPAQRL is that while being a language for querying RDF, it is not RDF itself. Also obviously, a content language for agent communication must support other types of communicative actions, for example, request for action. For these reasons we did not consider using SPARQL as such. Rather, when designing S-APL we included into it features analogous to most of the SPARQL's
ones.

The beliefs storage of an S-APL agent can be queried externally by other agents, of course subject to security and other policies. The core of a query is the same as if the agent itself would query its beliefs to check the premises of a rule. The core of the query has to be wrapped with sapl:I sapl:want { {sapl:You sapl:answer {..query..} } }. The use of "sapl:I sapl:want" may look unnecessary. However, this allows distinguishing between sapl:I sapl:want {...} and e.g. :Boss sapl:want {...}, i.e. mediating a wish of another agent. Both cases may require exactly the same action to be taken, however, may affect differently on whether the agent will comply or not.

As the response, the agent is to send the matching part of its belief storage, or, if no match, the query itself wrapped with sapl:I sapl:doNotBelieve {...}. Below, we list two small S-APL programs that an agent has to load in order to be able to be queried this way. The first one, Listener.sapl, instructs the agent to continuously wait for incoming messages marked with"SAPL" ontology. The additional rule of the program adds for every incoming request an additional existsWhile statement so that the request is removed after 5 seconds if no rule has taken it for processing.

```
/*Listener.sapl*/
{sapl:Isapl:dojava:ubiware.shared.MessageReceiverBehavior}
sapl:configuredAs {p:matchOntology sapl:is "SAPL".
p:waitOnlyFirst sapl:is false}.
{{{?requestID p:received *} sapl:ID ?id. sapl:Now sapl:is ?time.
sapl:I sapl:doNotBelieve
{?x sapl:existsWhile *. ?x sapl:hasMember ?id}
} => {
{?id sapl:is sapl:true} sapl:existsWhile
{sapl:Now sapl:is ?newtime. ?newtime < ?time+5000}}
} sapl:is sapl:Rule
```

2.4 PROFI: Proactive Future Internet

Big industrial players involved in the Future Internet technology area are interested in seeing Future Internet platform self-manageable, in particular, in the aspects of optimization, maintenance, performance management, and reconfiguration[21]. The PROFI technological concept (as further elaboration of the SmartResource, and UBIWARE [18] concepts developed by the Industrial Ontologies Group) is seen as a promising approach to cope with self-manageability problem in its versatility. As systems (inter alia networking) become increasingly complex, traditional solutions to manage and control them reach their
limits and pose a need for bringing self-configuration and\ self-management aboard. Also, heterogeneity of the ubiquitous components, communication standards, data formats, networking protocols, etc., creates significant hassles for interoperability in such complex systems. The promising technologies to tackle these problems are the SemanticWeb for interoperability, and Software Agents for management of complex systems.

The major PROFI objective is to provide the basis for such future Internet overlay architecture that will integrate autonomous (self-managed) proactive programmable Internet components. To achieve that, a specialized agent-driven middleware platform [19] is to be designed. It is envisioned that each future Internet programmable component, e.g., host, router, edge cluster, edge node, etc. (terms are taken from the

GENI vision [20]) will be assigned a representative agent within PROFI. The resulting multi-agent system will be the core of the targeted future Internet overlay architecture for enabling flexibility, adaptability, self configurability and self-management of the future Internet infrastructure. Utilization of semantic technologies in PROFI will ensure efficient and autonomous coordination among PROFI agents and will thus bring another dimension to interoperability of future Internet components and entities.

Also, Future Internet Upper Ontology will be designed as an important asset contributing to interoperability realization within Future Internet platform. FI Upper Ontology will be used not only for the benefit of PROFI middleware architecture, but also and most importantly for facilitation of interoperability and integration of existing and new future components and solutions. This implies that FI Ontology will also be used to cope with problems other than specific PROFI issues, such as naming and addressing, interoperability and integration, security, privacy and trust on the scale of the entire future Internet architecture. The PROFI will enable various information and networking components to automatically discover each other and to configure a complex system functionally composed of the individual components' functionalities.

PROFI can be considered as an engine for declarative networking. PROFI will enhance available declarative networking languages by adding to them explicit semantics (according to the W3C standards) specified in the ontological format. Our Semantic Agent Programming Language (S-APL), which is an RDF-based language for declarative programming of proactive components, can be utilized within the PROFI platform. S-APL is suitable for semantic description/annotation of various physical (e.g., network components, devices, etc.) and virtual (e.g., informational facts, rules, policies, commitments, individual and collaborative behaviors, etc.) resources. In S-APL, there is no strict separation between the data (descriptive knowledge) and program code (behavioral knowledge). S-APL is assumed to be used both as the programming language (for specification of network components behavior) and a communication\ content language (among architectural components). The syntax for RDF used in S-APL is one of Notation3\ (N3), which is more compact than RDF/XML. S-APL is a hybrid of semantic rule-based reasoning engines such as CWM (http://www.w3.org/2000/10/swap/doc/cwm) and agent programming languages (APLs). From the semantic reasoning point of view, S-APL is CWM extended with common APL features such as the Beliefs-Desires- Intentions architecture, which implies an ability to describe goals and commitments among the overlay architectural components – data items presence of which leads to some executable behavior, and an ability to link to sensors and actuators implemented in a procedural language. From the APL point of view, S-APL is a language that has all the features (and more) of a common APL, while being RDF-based and thus providing advantages of semantic data model and reasoning. S-APL introduces the semantic cognitive agent architecture, which has three layers: the toplevel Behavior Engine, the middle-level S-APL storage, and the bottom-level Reusable Atomic Behaviors (RABs) and the blackboard (for non-semantic data). The architecture also enables agents to access both S-APL programs/data and RABs from remote repositories.

3 Semantic descriptions of resources with Proactive behavior of autonomous condition monitoring applications

Our intention is to make industrial devices (as well as other Semantic Web Resources) proactive in a sense that they can analyze their state independently from other systems and applications, initiate and control own maintenance proactively.

The main idea of the new approach is that a software agent is assigned to each word of text under consideration with the help of semantic web middleware. Agents have access to a comprehensive repository of knowledge about possible meaning of words in the text and engage into negotiation with each other until a consensus is reached on meanings of each word and each sentence. To simplify the process of extracting meanings, the method performs an initial morphological and syntactic analysis of text.

Web Services should be able to interpret the information that they receive. Autonomous Web Services not only minimize the human intervention by automating interaction with other Web Services, allowing programmers to concentrate on application development, but also are able to recover from failures more efficiently by automatically reconfiguring their interaction patterns.

4 CONCLUSION

In this paper we have proposed an enhancement in middleware architecture for enabling flexible, autonomous interaction between Semantic WS and agent services. We have also highlighted the technologies and how they come together in order to achieve the whole process. For now we have considered the semantic web middleware where an agent negotiates with the web service. An initial implementation of this architecture has been done and we intend to improve the proposed design so as to cater more negotiation protocols especially, the auction protocols in future. We expect that this initial effort of conducting negotiation via Gateway service bridging agents and WS is only a prelude to exploring the immense potential it offers as a means to compose, invoke, administer and manipulate heterogeneous service populations in future.

ACKNOWLEDGMENT

I would like to acknowledge the ongoing support of my parents and my family members, whose patience and encouragement during these long days and night have been paramount in making this research paper a reality.

References

[1] Ermolayev, V., Keberle, N., Plaksin, S., Kononenko, O., & Terziyan, V. (2004). Towards a framework for agent-enabled semantic Web service composition.

[2] Kaikova, H., Khriyenko, O., Kononenko, O., Terziyan, V., & Zharko, A.

(2004).
Proactive self-maintained resources in Semantic Web. Eastern-European Journal of Enterprise Technologies, 2(1), 4-16.
[3] Khriyenko, O., & Terziyan, V. (2004, June 24-26). OntoSmartResource: An industrial
resource generation in semantic Web. In R. Schoop, A. Colombo, R. Berhardt,& G. Schreck (Eds.), Proceedings of the Second IEEE International Conference on Industrial Informatics (INDIN '04), Berlin, Germany (pp. 175-179).
[4] METEOR-S: Semantic Web Services and processes. (2005). LSDIS Lab, University of Georgia. Retrieved from http://lsdis.cs.uga.edu/ and http://swp.semanticweb.
Org
[5] Semantic Web Services Initiative/Semantic Web Services Architecture. (2005).
Retrieved from http://www.daml.org/services/swsa
[6] T. Berners-Lee, Notation 3: An RDF language for the semantic web. Tech. rep.,
WorldWideWeb Consortium (W3C), 2000.
[7] Z. Ding, Y. Peng, R. Pan, Y. Yu, A Bayesian methodology towards automatic ontology mapping, in:Workshop on Context & Ontologies, Twentieth Conference on
Artificial Intelligence (AAAI), 2005.
[8] A. Doan, A. Halevy, Semantic integration research in the database community,
a brief survey, AI Magazine 26 (1) (2005) 83–94.
[9] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Learning to map between ontologies on the semantic web, in:WorldWideWeb, ACMPress, 2002, pp. 662–673.

[10] The DAML Services Coalition. DAML-S: A Semantic Markup For Web Services, http://www.daml.org/services/daml-s/2001/10/daml-s.pdf.
[11] Ankolekar, A., Huch, F. and Sycara, K. Concurrent Execution Semantics for DAML-S with Subtypes. In The First International Semantic Web Conference (ISWC), 2002.
[12] McIlraith, S.A., Son, T.C. and Zeng, H. Mobilizing the Semantic Web with DAMLEnabled Web Services. In Semantic Web Workshop, 2001.
[13] Agent-based information services for process automation http://automation.tkk.fi/files/proage/
[14] http://www.cs.jyu.fi/ai/papers/ICSC-2008.pdf
[15] T. Berners-Lee. Notation 3: A readable language for data on the Web. Online:
http://www.w3.org/DesignIssues/Notation3.html.
[16] T. Berners-Lee, D. Connoly, L. Kagal, Y. Scharf, and J. Hendler. N3Logic: A logical framework for the World Wide Web. Theory and Practice of Logic Programming, 8(3):249–269, 2008.
[17] Foundation for Intelligent Physical Agents. Agent Communication Specifications. Online: http://www.fipa.org/repository/aclspecs.html.
[18] University of Jyväskylä. UBIWARE: Smart Semantic Middleware for Ubiquitous Computing. Online:
http://www.cs.jyu.fi/ai/OntoGroup/UBIWARE_details.htm.
[19] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Terziyan. Smart semantic middleware for the internet of things. In Proc. 5th Intl. Conf. Informatics in Control, Automation and Robotics (ICINCO'08), Volume ICSO, pages 169–178, 2008.
[20] GENI. Global Environment for Network Innovations. Online: http://geni.net/.
[21] Proactive Future Internet: Smart Semantic Middleware for Overlay Architecture Vagan Terziyan, Dmytro Zhovtobryukh, and Artem Katasonov