

# Agreement-aware Semantic Management of Services

Qian Zhao, Yu Zhou, Mark Perry  
University of Western Ontario  
{ qianzhao@csd|yuzhou@alumni|markp@csd}.uwo.ca

## Abstract

*Automation of versatile process management will aid greater efficiency and lower costs. It is widely expected that highly structured agreements should be executed and enforced automatically. In the progress of building an executable license agreement framework, we have noticed that diverse agreement processing and automation have commonalities leading us to propose a generic architecture that can provide various automated agreement services that leverage ontology, rule and agent technologies.*

## 1. Introduction

Enterprise-oriented IT service companies, like IBM, HP and Sun, have been promoting 'On Demand Business' [1], 'Adaptive Enterprise' [2] and 'Utility Computing' [3]. The ideas of all these concepts are similar, that is to provide computing services in the 'user consumption' model. This kind of pricing strategy is widely effective in many traditional utility services like water, hydro, etc. In the IT service industry, buyers typically pay a uniform cost upfront, even though usage may vary significantly with varying environments. Some customers think they cannot afford the high upfront price, others feel money is wasted if the software service is under-used. Intense competition requires businesses to depend upon advanced IT services with improved effectiveness, efficiency and customer convenience. On the other hand, cost containment is vital for survival. IT vendors are adopting the 'pay as you go' model for lower priced, on demand services for subscription, to attract more customers. On demand services pricing is based on usage and tied to the quality of service.

The feasibility and success of these services depend on highly automated management mechanisms to calculate usages, maintain the quality, etc, with Autonomic Computing (AC) being proposed as their backbone. AC has four basic characteristics [4]: self-configuring, self-healing, self-protecting, and self-optimizing, i.e. they are self-managing. At the

heart of AC is the Autonomic Manager (AM) [4], working like the neural system in human being, in which there are sensors to monitor the states of service(s); there are effectors to control the service(s); and most importantly, there is a knowledge base for analysis and planning.

Subscription to an on demand service requires agreements, such as a license and Service Level Agreement (SLA), to make sure customers know their rights and constraints. Other types of consensus between the customer and vendor, and/or end users that regulate the usage internally, are also agreements.

Enforcing these agreements requires flexible management mechanisms such as the AM. However, when we look into the structure of AM, we find four major challenges:

- How to represent agreements and other information as knowledge?
- How to analyze collected service status against the knowledge?
- How to monitor the service?
- How to plan and enforce the impact if necessary?

This research focuses on the first two challenges as there are already some potential solutions to the last two challenges, including Tivoli monitoring and controlling agents [5]. There is little work directly addressing knowledge representation and information analysis in the service management area.

## 2. Approaches

There is more than one possible way to represent agreements as knowledge. In searching for the appropriate representation of agreements, we take several key objectives into consideration:

- Machine readable: essential to automate agreement enforcement.
- Human readable: useful for human comprehension to reduce the obstacles faced by domain experts.
- Arbitrary structure: agreement formats vary, depending on the application domain. The representation shall support diverse agreement structures without too many limitations.

- Interoperability: diverse management components understand the agreements and communicate with each other.
- Intelligence: represented knowledge may add intelligence to agreement enforcement.

### 2.1. Plain Text and Structured Agreements

The most common representation of agreements is still plain text, and existing expertise can be leveraged, but this type of agreement is error-prone and difficult to automate. Technical and vague terms and varying structure, demand complex compilers or interpreters that have little reusability. Varying terminologies hinder the communication between two agreement interpreters. It is hard to detect typing errors, conflicts, and irrational statements in plain text agreements.

Structured agreements are easy to process, manage and enforce. However, they require extra effort to define the structure, and lawyers writing the agreements need help from developers. They can aid automation, and overcome the plain text drawbacks.

### 2.3. Ontology to describe structured agreement

The concept of ontology originates from the social science, “the science of being or reality in the abstract, particularly as related to ideas or theories” [6]. It was adopted in Artificial Intelligence as “specification of a conceptualization of a knowledge domain” [7]. Ontology, as a formal representation of shared understanding, is useful in describing structured agreements. Reasoning upon ontology and emerging ontologies is an asset to make management more intelligent and to overcome communication barriers.

### 2.4. When AC meets the Semantic Web

Ontology in the context of knowledge representation has been taken up in other IT fields, e.g. “on the Web, the term (ontology) applies to the many ongoing efforts to develop topic-specific sets of XML-friendly language, rules and definitions” [8]. Such contributions to the Semantic Web [9] aid transformation of the web, and exemplifies an outstanding ontology application. The ontologies written in RDF (Resource Description Framework) [10] and OWL (Web Ontology Language) [11] form the basis of Semantic Web, which provides machine readable intelligence coming from hyperlinked vocabularies from the Web, so that software agents can understand web content in a way that is comparable to human understanding. As on demand services are delivered over the Internet or

intranets, it is promising to have management knowledge published as ontologies.

Once expressed in ontology languages of semantic web, like RDF and OWL, the agreements and other management knowledge on which AMs rely become semantically meaningful and can be reasoned upon directly. The AM itself becomes semantics-enabled and could be part of the Semantic Web. With the help of ontology, the Semantic AM can distribute its tasks easily among multiple intelligent agents to balance the workload and to speed up the processing. Further more, web services would become semantic services, making the Semantic Web more powerful, should they express their discovery, description and interaction as ontology and be understood and used by agents. Ideally, all agents will work harmoniously under a global schema, each being easily replaceable and interchangeable.

## 3. The SmArt Model

We start with the **Semantic Agreement (SmArt)** service in a controlled scope – a hosted data center that only provides services over an intranet or to an extranet via Virtual Private Networks – so the user base, with limited variations, is relatively stable.

The data center is hosted by a service provider that allows customers to build and subscribe to the services they want. The service is a combination of hardware, software and licenses, which provide functionalities for the user. At subscription, the detailed license, service level, rate, etc are specified in different agreements. By signing onto these agreements, the customer agrees to the service terms and the vendor promises to maintain the assured quality of service, e.g. the customer pays as agreed. There will be license agreements, SLAs, and even reservation schedules or internal administration agreements within the customer organization against which the concerned services are constantly confirmed.

The autonomic SmArt service works as shown in Figure 1. The Semantic AM consists of two main components, the SmArt Knowledge Core (SKC) and SmArt Agents (SAs).

- SKC, represented as ontologies and rules, includes the agreements and their definitions, the domain concepts and their relationships, and the constraints that are mostly as rules.
- SAs carry out tasks of the autonomic manager, including monitoring, analyzing, planning, controlling and so on.

When a request comes in, SAs consult the SKC to take corresponding actions: start up the SmArt service, monitor and analyze service status, or control the

service as necessary. The SKC is referenced by multiple SAs and serves as the communication basis. The service can adapt itself to changing situations with help of these SAs and the SKC.

1) The service user notifies the SmArt Service Factory the need of a particular service.

a) The SmArt Service Factory discovers the agreement instances bonded with this service and this user.

b) The Factory consults ontologies that tell how to configure the requested service, the agreement structure, as well as modeling the relations between the agreements. With the help from these "other ontologies" that include all other ontologies except that of agreement instances, the Factory is able to locate agreement instances and identify required resources.

c) The Factory may also need to consult some constraints that have impact on the demanded service, agreements and resources to make sure such a SmArt

b) The Analyzer consults "other ontologies" for concept correlation rules indicating how to do analysis upon data.

c) The Analyzer searches through constraints ensuring the service usage does not exceed the service provider's ability.

6) The SmArt Service Controller is notified if an action needs to be taken.

a) Ontology of agreement instances is referred to so as to identify which action needs to be taken.

b) The Controller gets information about how to apply the control from "other ontologies".

c) The Controller uses constraints to decide which concrete implementation shall be invoked to carry out the control.

7) The Controller exports controls to the SmArt service. The service is adjusted accordingly to function better.

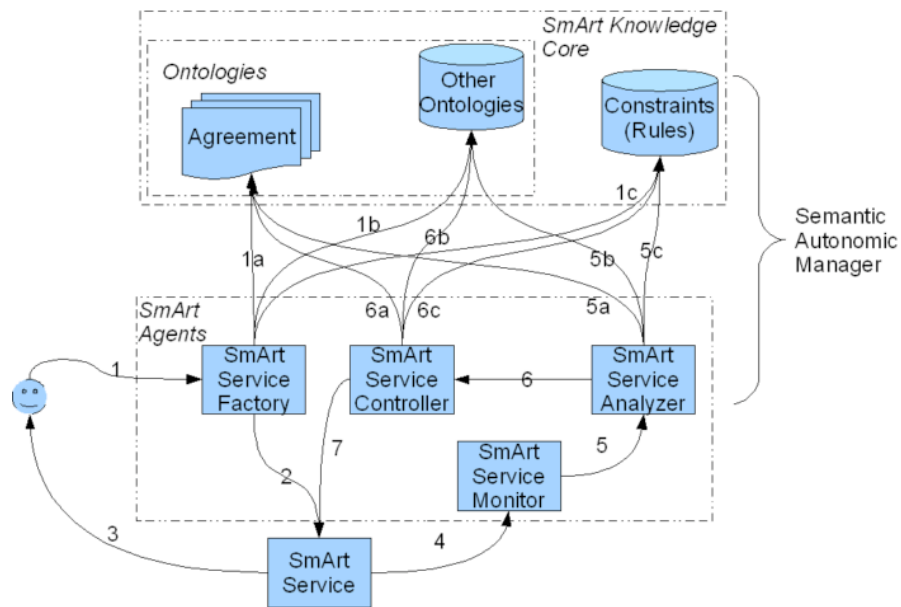


Figure 1 SmArt Model

service request is supportable.

2) After the service request is validated, either the SmArt Service Factory gets the required resources and brings the service to life, when no such service already exists or when the service is of monopoly, or the Factory finds where the service is running.

3) The running service exposes itself to user demand.

4) The running service is observed by the SmArt Service Monitor.

5) The SmArt Service Analyzer analyzes data collected by the Monitor and detects abnormalities.

a) The Analyzer checks with the agreement instances to search for violations against agreement provisions.

#### 4. SmArt Ontologies

Ontology consists of statements that describe the concepts and their relations. A basic ontology statement consists of three parts: subject, predicate, and object. Both the subject and the object are concepts, and the predicate is the relation between these two concepts. Using this kind of expression, knowledge can be represented formally and be queried with tools.

## 4.1. Ontology Types

From the model, we know that there will be many ontologies used in the SmArt service, such as definitions of agreements and other concepts (the meta-data), relationships among agreements, concepts and their elements (the semantics), agreement instances, etc. It is necessary to classify those ontologies for efficient management and effective acquisition.

In [12], Guarino proposed a way of classifying multiple ontologies. A step further from the commonly mentioned ontology types are upper ontology and domain ontology; Guarino's classification distinguishes knowledge of task and application workflow and thus presents a finer grained classification. The top level ontology, sometimes called the upper ontology, represents knowledge independent of any particular domain. The domain ontology and the task ontology depend on the top level ontology and model the knowledge of domain concepts and domain task respectively. The application ontology depends on the domain and the task ontology to represent knowledge of a particular application. Ontologies of SmArt service can be divided into these categories as well.

## 4.2. Domain Ontology

To develop the domain ontology, we first need to enumerate all concepts in a particular domain and specify every relationship among them. The CommonKADS [13] is a formal methodology for ontology modeling. Developers can have their own methods of doing so as long as the domain concepts and relations are well extracted and represented.

There are five basic concepts in the SmArt service. Their interactions are shown in Figure 2. In this diagram, classes are concepts and properties are predicates; the domain means the class being pointed at is the subject; and the range means the object. The service here is a SmArt service that satisfies some users' requests. To serve customers, the service needs to get some resources. These resources may impose some constraints that regulate the service and its corresponding agreements. Besides constraints, the service should not violate its agreements either.

We can then break down the five concepts and their relations further, which give more detailed information regarding each concept and introduces new concepts and new relations into the domain ontology. This kind of concept refining can keep going on and extract more classes such as the customer, the provider and specific resource types, and more predicates like a user

belonging to a sub group within the customer. The refining process stops when the granularity reaches a satisfactory level where finer-grained decomposition is unnecessary.

## 4.3. Task Ontology

Besides representing domain concepts as domain ontologies, we can also separate the general and reusable task logic from the implementation and express them as task ontologies. This means the Semantic AM is like a robot with a variety of features, each realized by a particular implementation, and the task ontology is the control unit. To make a task ontology working the Semantic AM must provide at least one feature implementation for each element in that task workflow. Once the control unit is loaded, the "robot" can carry out tasks accordingly with little or no human intervention.

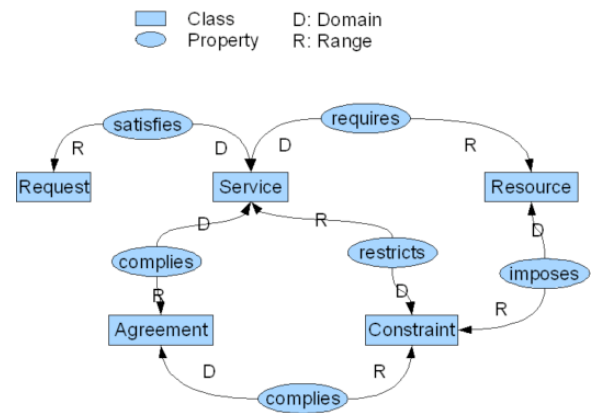


Figure 2 Basic SmArt Domain Concepts

Because of the separation of logic from implementation, the ontology of a task can be used repeatedly in diverse AMs although the concrete implementation of its workflow elements may differ each time. The separation also facilitates plug and play. A feature of an AM can be updated easily by replacing its implementation with another counterpart. The AM after replacing will work in the same way as it did before. Last but not least, the separation makes the implementation reusable. A feature of one AM can be employed by ontology of various tasks as long as it is recognizable and understood by them.

Regarding the concerned domain, the SmArt service, we may have a set of task ontologies, one of which is shown in Figure 3.

The ServiceConfigFactory needs to discover corresponding agreement(s) after receiving a request. The agreement(s) validate(s) the request with the

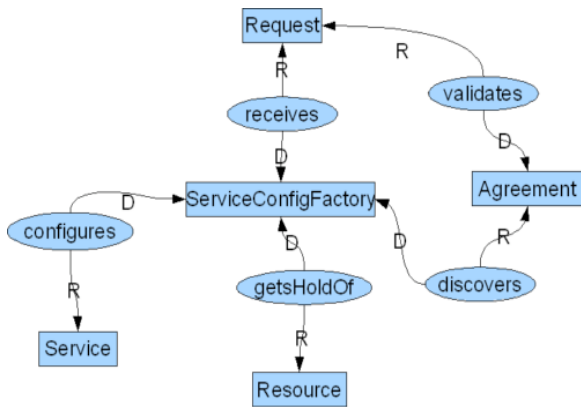


Figure 3 SmArt Task Ontology  
– Service Configure

precondition that they are associated with the request. The ServiceConfigFactory reserves resources for the requested service with the precondition that the request is valid. The ServiceConfigFactory configures and start the service to serve customers with the precondition that those resources are available. The precondition for each predicate ensures the order of execution, although task ontologies do not give the order explicitly. As with the domain ontology, the task ontology can also be refined until the desired granularity is achieved.

## 5. SmArt Agents (SAs)

After the task ontology has been extracted, embodiment of the task ontology is left alone and it may be implemented by one or more SAs. Other SAs include features of the "robot" mentioned in the task ontology section. In SmArt services, SAs follow different task ontologies and work together as the semantic autonomic manager. Some SAs already shown in the model are:

- Service Factory is responsible to configure and de-configure and start/stop the SmArt service, which could be part of the Service Controller.
- Service Monitor is to collect status information of the SmArt service and to document the data.
- Service Analyzer runs analysis on collected service data to detect abnormalities.
- Service Controller is to apply control on the SmArt service when necessary.

Other classes in the task ontologies are implemented as SAs as well.

Predicates are implemented as methods of their agent-incarnated classes. Any SA may have several implementations, and even for one SA any method may be realized in multiple ways. An SA finds the answer to the question: which implementation shall be used within a given application from application ontologies?

To leverage ontologies, an agent shall possess four basic capabilities: query the domain ontology, query the task ontology, query the application ontology, query an agent from the agent registry and invoke it.

The SAs could be traditional programs, scripts or a combination of the two. Regardless of the code they use, SAs communicate based on concepts defined formally in ontologies so that they can understand each other despite their diverse platforms and use of different programming languages.

On the other hand, SAs can be implemented without help from task ontology. In this case, the task logic is bound to the ad hoc implementation, which is straightforward but to some degree loses reusability.

## 6. Implementation

We started our research using semi-structured eXtensible Markup Language (XML), in which we defined an XML Schema to represent the structure of agreement and made the agreement interoperable among programs from various platforms. We soon encountered some obvious limitations on XML-based representation, e.g. lack of meaning, needs of ad-hoc interpretations, inability of incorporating intelligence and reasoning, etc. However, the accumulated knowledge on agreement management and enforcement that is captured in the XML Schema, as well as in related ad-hoc Java based agreement enforcement, became the foundation of this research.

We kept a close look at the development of the Semantic Web after RDF and OWL became W3C recommendations, and successfully transferred most of our previous research results from XML Schema to ontologies in RDF/OWL with enhanced business logic captured in the task ontology. Meanwhile, we also surveyed the Foundation for Intelligent Physical Agents (FIPA) specification [14] and several multi-agent system (MAS) implementations with FIPA compliance, e.g. JADE [15] and ABLE [16] along with their rationale with AC.

The research behind this paper, based on our previous research on Policy Enforcement [17], Policy-based License Management [18], and various surveys on Semantic Web and Agent technologies, is being implemented as a relatively generic agreement

management and enforcement framework, using OWL, Jena [19], and ABLE.

## 7. Conclusion and Future Work

In this research, we focus on exploring the possibilities of using ontology and agent technologies to provide a flexible solution for agreement management and enforcement. The early results from our experiments are encouraging. For the first time, we are able to separate the domain knowledge and task logic in this area from ad-hoc implementation. In other words, the SmArt model and corresponding framework will be able to manage and enforce various agreements even after the deployment by using SmArt Agents to interpret and execute the SmArt Ontologies.

In the future, we will push our research forward in several frontiers:

- Complete the SmArt Domain Ontology;
- Identify the SmArt Task Ontology in more detail;
- Design and implement the more flexible SA framework that is associated with SmArt Task Ontologies;
- Apply SmArt architecture and framework in real world scenarios and evaluate the efficiency and usefulness.

## Acknowledgements

The authors thank the Natural Science and Engineering Research Council of Canada, and the IBM Centre for Advanced Studies.

## References

- [1] J. Hagel, "Becoming an On Demand Business: Innovation Breaks New Ground", On Demand Business White Paper, International Business Machines Corporation, [http://www-306.ibm.com/e-business/ondemand/us/pdf/OnDemandInnovation\\_9.pdf](http://www-306.ibm.com/e-business/ondemand/us/pdf/OnDemandInnovation_9.pdf), Feb. 2005
- [2] Hewlett-Packard Company, "Infrastructure and Management Solutions for the Adaptive Enterprise", Adaptive Enterprise White Paper, Hewlett-Packard Company, [http://www.hp.com/products1/promos/adaptive\\_enterprise/pdfs/vision\\_for\\_ae.pdf](http://www.hp.com/products1/promos/adaptive_enterprise/pdfs/vision_for_ae.pdf), 2003
- [3] S. Wells, "Grid Power", Executive Boardroom Program - Boardroom Minutes, Sun Microsystems Inc., <http://sun2.delivery.net/sun/sunBR/0605feature.jsp>, Jun. 2005
- [4] B. Miller, "The autonomic computing edge: Can you CHOP up autonomic computing?" Autonomic Computing on IBM developerWorks, International Business Machines Corporation, <http://www-128.ibm.com/developerworks/autonomic/library/ac-edge4/>, Jun. 2005
- [5] IBM, "IBM Tivoli Monitoring - Product Overview", Tivoli Product, International Business Machines Corporation, <http://www-306.ibm.com/software/tivoli/products/monitor/>, Feb. 2006
- [6] P. L. Greaves, Jr., "Mises Made Easier Glossary for Human Action - Ontology", <http://www.mises.org/easier/O.asp#9>, 1974
- [7] G. M. Browne, and J. Jermey, "Website Indexing 2nd Edition: Glossary - Ontology", <http://www.webindexing.biz/Webbook2Ed/glossary.htm>, 2006
- [8] D. Moschella, "Some IT Terms Really Are Greek", Computer World, <http://www.computerworld.com/managementtopics/ebusiness/story/0,10801,70557,00.html>, Apr. 2002
- [9] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web - a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities", Scientific American, May 2001
- [10] G. Klyne, and J. Carroll eds., "Resource Description Framework (RDF): Concepts and Abstract Syntax", W3C Recommendation, <http://www.w3.org/TR/rdf-concepts/>, Feb 2004
- [11] P. F. Patel-Schneider, P. Hayes, and I. Horrocks eds., "OWL Web Ontology Language Semantics and Abstract Syntax", W3C Recommendation, <http://www.w3.org/TR/owl-absyn/>, Feb 2004
- [12] N. Guarino, "Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration", Lecture Notes in Computer Science, Springer Verlag KG, Germany, 1997, issue 1299, pp. 139-170
- [13] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. V. de Velde, and B. Wielinga, "Knowledge Engineering and Management - the CommonKADS Methodology", the MIT Press, Dec 1999
- [14] FIPA Standards Organization, "FIPA Specifications", The Foundation for Intelligent Physical Agents, <http://www.fipa.org/>, 2005
- [15] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "Java Agent Development Framework: a white paper", EXP (Special issue on JADE of the TILAB Journal), <http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf>, Sep. 2003, V.3 n.3 pp. 6-19
- [16] M. Meyer, "The features and facets of the Agent Building and Learning Environment (ABLE)", Autonomic Computing on IBM developerWorks, International Business Machines Corporation, <http://www-128.ibm.com/developerworks/autonomic/library/ac-able1/>, Oct. 2004
- [17] Y. Zhou, Q. Zhao, and M. Perry, "Policy Enforcement Pattern", Proceedings of Pattern Language of Program, USA, Sep. 2002
- [18] Q. Zhao, Y. Zhou, and M. Perry, "Component Software and Policy-Driven Licensing Model", Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Italy, Jun. 2003
- [19] B. McBride, "Jena: Implementing the RDF Model and Syntax Specification", Semantic Web Workshop, WWW2001, <http://www.hpl.hp.com/personal/bwm/papers/20001221-paper/>, 2001