Western University
Scholarship@Western

Computer Science Publications | Computer Science

2008

An Ontology for Autonomic License Management

Qian Zhao
University of Western Ontario, qianzhao@csd.uwo.ca

Mark Perry
University of Western Ontario, mperry@uwo.ca

Follow this and additional works at: http://ir.lib.uwo.ca/csdpub

 Part of the Computer Sciences Commons, and the Contracts Commons

Citation of this paper:
Zhao, Qian and Perry, Mark, "An Ontology for Autonomic License Management" (2008). *Computer Science Publications.* Paper 8.
http://ir.lib.uwo.ca/csdpub/8

An Ontology for Autonomic License Management

Qian Zhao and Mark Perry
Department of Computer Science
University of Western Ontario
London, ON, Canada
Email: {qianzhao|markp}@csd.uwo.ca

Abstract

The license agreement can be seen as the knowledge source for a license management system. As such, it may be referenced by the system each time a new process is initiated. To facilitate access, a machine readable representation of the license agreement is highly desirable, but at the same time we do not want to sacrifice too much readability of such agreements by human beings. Creating an ontology as a formal knowledge representation of licensing not only meets the representation requirements, but also offers improvements to knowledge reusability owing to the inherent sharing nature of such representations. Furthermore, the XML-based ontology languages such as OWL (Web Ontology Language) can be user friendly for the non-developers who are often those responsible for implementing and managing such license agreements. This paper shows our use of ontology to represent the license agreement in a development prototype. The ultimate goal is to build ontology for the license management domain that will facilitate autonomic knowledge management. Knowledge based on such ontology can then be shared and utilized by many types of license management system.

1 Introduction

To provide a high quality service, a modern data center has to allow the customer to subscribe a service configuration that best fits their preferences, and responds to any customer request whenever it occurs. Accordingly, the customer promises to assume certain responsibilities that come with these benefits. Both the benefits and responsibilities are specified formally by various agreements, and for any significant system such agreements are formally signed by both the customer and the service vendor. While the data center will keep to the service level agreement (SLA) that defines the levels of service quality it has to maintain, it also wants to make sure that the customer complies with the license agreement (LA) that define how the subscribed service should be used. This means a license management system is needed to govern the constrained services during their life cycle to avoid service abuse, and the license agreement is where the knowledge of such a management comes from. To achieve higher efficiency (lower cost, faster, and accurate), the license agreement has to be automatically interpreted and enforced in the management system. Such a process is called license agreement automation. Two major challenges related to the agreement representation that need to be addressed in such automation are:

- Representing license agreements in a machine readable format without sacrificing too much readability by humans; and
- Ensuring that each agreement is understood by the automation processes without ambiguity.

In our previous work on agreement automation we proposed the SmArt (Semantic Agreement) [1] framework to facilitate automatic agreement interpretation and enforcement. The SmArt system was designed to leverage ontology to represent license agreements and other shared knowledge, and to employ agents to carry out tasks [2]. Here we describe the other parts of the system to build a prototype to manage a simple software service license. A license management system based on the SmArt Framework operates in the way shown in Figure 1.

Ontology itself is a formal representation of knowledge widely used to aid electronic data processing. Common examples include cyc [3], and those in biological and medical areas like gene ontology [4] and foundational model of anatomy [5]. Ontology uses a set of tuples to describe some knowledge in a certain domain and these tuples are easily processed by computers. With the help of recent markup ontology languages like OWL/RDF, ontology has become more friendly for human understanding than traditional ones like CycL [6]. Meanwhile, as ontology defines all necessary concepts (classes) and their relations in

0-7695-3093-1/08 $25.00 © 2008 Crown Copyright
DOI 10.1109/ICAS.2008.12

IEEE computer society

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

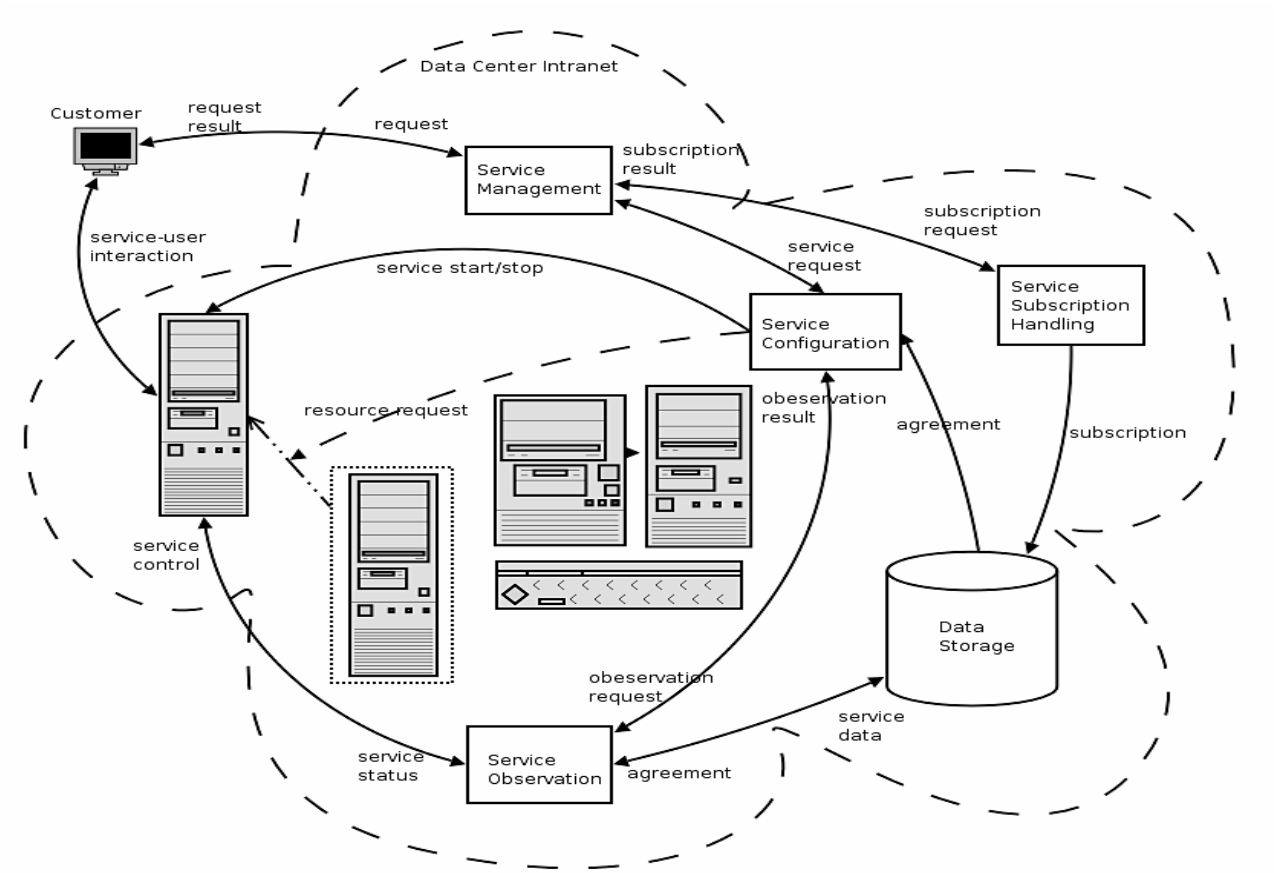

Figure 1. An overview of the SmArt system.

a knowledge domain as it subject-property-object tuples, computer programs can easily "understand" the semantics of the defined concepts without ambiguity. For example, if we have *John-isA-prof* and *prof-worksIn-univerisity*, as the ontology, a program committed to this ontology recieves *John-worksIn-uwo* as input and understands "*uwo* is a *university*" without having this piece of knowledge hardcoded into its logic. No matter how many programs there will be, as long as they are committed to the same ontology, these programs all understand "*uwo* is a *university*" and act according to this fact.

We can see that the ontology is well suited to meet the two representation challenges of license agreement automation. Applying the SmArt framework design to the domain of license management gives us a management system that is able to automate license agreemments, and is shown here. It should be noted that the management of licenses is only an example application of SmArt.

The foundation of the SmArt framework is its ontology based knowledge core, and such a knowledge core is therefore an essential part in the system for license management as well. To build this ontology, hence to build the knowledge core, concepts and relations in the license management domain have to be carefully analyzed, organized and selected. This paper discusses ontologies of such a license management system in detail. Section 2 gives the overview of the license management system that is based on SmArt framework and able to automate license agreements. Section 3 discusses the domain ontology of license management, focusing on the classification of concepts and vocabularies. Section 4 introduces the task ontology that expresses reusable tasks, represented as some relations each of which consists of an "actionable" property between two concepts. Section 5 describes the development environment, and this is followed by the last section which summarizes the paper and future work.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

2 System Overview

We need to first discuss what kind of licenses are used in the typical software service center. Based on our early work in [7] [8] [9] and [10] as well as discussion with software developers who work on data center management systems, we identified five proprietary license types as the most common ones in a data center out of fifteen basic licenses. The five licenses are the time-based, the concurrent, the consumptive, the cumulative, and the named license.

- **Time based license**: license is only valid between the start time and end time; typical in service subscription because most subscriptions are based on time.
- **Concurrent license**: there is a limited number of licenses available at any given time during the subscription period; however, these licenses are reusable, which means a license can be reassigned to another user request after it is returned by previous request.
- **Consumptive license**: there are a certain number of licenses available throughout the whole subscription period; these licenses are not reusable, which means one license cannot be shared with multiple users.
- **Cumulative license**: non-reusable licenses with no limit throughout the subscription period; although this type of license is always available due to the no-limit characteristic, the license usage is recorded and calculated for later reference, which is how the pay-per-use works. There may be a threshold in this license type that some actions, like notifying the service subscriber with a usage report, can be taken when the threshold is achieved.
- **Named license**: license is only valid to a certain group of users; every time a user requests such a license, a pre-defined list of user names is checked to see if this user is included.

After identifying these main license types, their properties need to be articulated in the license agreements for future reference.

The data center intranet illustrated in Figure 1 is a protected network and we assume each customer outside this protected cloud has been identified and authenticated before its request is processed. We are not addressing the authentication/identification nor security issues here. The rectangles represent different component subsystems of management with their functionalities specified.

A customer interacts remotely with the service management component over the network. Before being able to use any service, the customer has to send a request for service subscription. Once this request arrives at the management component, the component identifies it as a valid subscription inquiry and asks the subscription handling component to respond. After the subscription is made and stored, the customer can use the subscribed service by simply sending a request. In the same way as in the subscription handling, the management component identifies the inquiry as a valid service request and asks the configuration component to bring the service up. The configuration component verifies the service request with its associated license agreement, makes the request to necessary resources, such as moving a server out of an idle resource pool or initiating a server instance, and then starts the requested service. Before the customer is allowed to use the service, the configuration component asks the observation component to observe service usage. The observation component keeps a close eye on the service status during the session of customer-service interaction, and controls the service as necessary according to the associated license agreement. When the interaction finishes, observation also ends, the service data is updated and the service is made ready for the next use.

Once the LA and license management processes are identified, they are represented as ontologies, LA in domain ontology and the processes in task ontology. The following two sections explain them in detail.

3 License Agreement Presentation: Domain Ontology

In the SmArt system, ontology is used to represent knowledge that can be shared among multiple agents and/or programs. One important part of knowledge is representing concepts in the license control domain. Domain ontology is the result of taxonomy. As noted in [1], there are five basic concepts: Agreement, Service, Resource, Constraint, Request. Below we will define these and other related concepts one by one.

A generic agreement structure is shown below, without the details of the agreement content. This section extends this generic structure into a functional license agreement template that is also extensible.

- Head
  - Service
  - Customer
  - Vendor
- Body
  - a list of ProvisionItem

When extending this generic structure into a specific agreement, the **Head** part can remain the same but the abstract ProvisionItem inside **Body** has to be replaced by the

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

concrete agreement provision item with further details per individual agreement type, such as, the ProvisionItem is replaced by the SLOItem in an SLA (Service Level Agreement) [11] [12].

For a license agreement, the provisionItem is replaced by the LAItem that is a provision of a license agreement. Due to limited space, the definition of LAItem and other concepts are given using the Backus-Naur form (BNF). As these definitions are transferred into OWL/RDF, anything in italic is an XML data type that needs no further defining, and is the object in the ontology tuple; any phrase beginning with a lower case letter generally is treated as the property in the tuple, while those quoted in angle brackets will be subjects and objects.

<LAItem>::= <LATerm>|<LicenseTerm>
<LATerm>::= <LicenseTerm>|<LATerm><AggregatingOperator><LATerm>
<AggregatingOperator>::= and |or
<LicenseTerm>::= <LicenseTermMetrics>[then {<Action>}
[else {<Action>}]]
<LicenseTermMetrics>::=
<TimeMetrics>|<ConcurrentMetrics
>|<ConsumptiveMetrics>|<CumulativeMetrics>
<TimeMetrics>::= timeSpan [start date end date]
<ConcurrentMetrics>::= concurrentMax [nonNegativeInteger
<Unit>]
<ConsumptiveMetrics>::= consumptiveMax
[nonNegativeInteger <Unit>]
<CumulativeMetrics>::= cumulativeThreshold
[nonNegativeInteger <Unit>]
<Unit>::= numberOfUsers |numberOfRequest |<TimeUnit>
<TimeUnit>::= year |month |week |day |hour |min |sec
<Action>::= stopService |generateReport |updateRegistry

In this example, the LicenseTermMetrics only includes four very common proprietary license types; the named license type, though most common, is omitted because the relation between service, customer and agreement in the Head part of an agreement realises naturally the named license in ontology. Any user has to belong to a Customer that is actually a group of users to be able to use a service subscribed by that Customer. So there is no need to define the named license and implement it again. At the same time, should there be a new license type emerging, it is easily accommodated by adding definition of the new term and including it in the LicenseTerm definition.

Similarly, the other basic domain concepts include Service, Resource, Request and Constraints, which be shown in BNF, although it should be noted that Constraint is a restriction bonded with one or two simple resources. It may or may not invoke some actions. In this research, the constraint is only associated with simple resources for simplification and clarification. If a restriction on aggregated resources is needed, it has to be decomposed to a set of constraints of consisting simple resources. Restrictions on services should be expressed as agreement provisions (**ProvisionItem**).

In the same way as defining the structure of a license agreement, the above definition for the other domain concepts can also be easily expanded to accommodate new items, or refined to adapt to other situations and meet different criteria. The definition is then expressed in OWL/RDF, making the license agreement and other domain concepts part of the ontology that makes up the SmArt Knowledge Core (SKC). The use of OWL/RDF based ontology also facilitates the extensibility of agreement definitions and other concept definitions.

4 License Management Processes: Task Ontology

The relation "*an agreement regulates a service*" is part of a task ontology where *regulates* is an actionable property that may represent a series of simple operations between concept *agreement* and concept *service*. As can be seen in the system overview shown in Figure 1, the SmArt system has four task ontologies that correspond to the four component subsystems respectively: Service Management, Service Subscription Handling, Service Configuration and Service Observation.

- **Service Management** works as the interface between customers and the SmArt system. It dispatches different customer request to corresponding functional component inside the SmArt system.
- **Service Subscription** handles service subscription request from customers and updates the system knowledge core every time a subscription is made.
- **Service Configuration** responds to a user's request of service. It verifies the request against relate license agreement before bringing the service into the ready-to-serve state.
- **Service Observation** monitors a service, analyzes its usage data and controls the service according to the related license agreement, preventing service abuse and generating usage report.

4.1 Service Management

The service management component accepts customer requests and works as the mediator between a customer and the system. It is the top level task ontology and guides how the whole system works. It contains two key ontology triples:

- Request asks to subscribe a service.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

- Request asks to use a service.

The concept request can be further distinguished as a subscription request or a service request, and thus the task ontology of service management is shown in Figure 2.

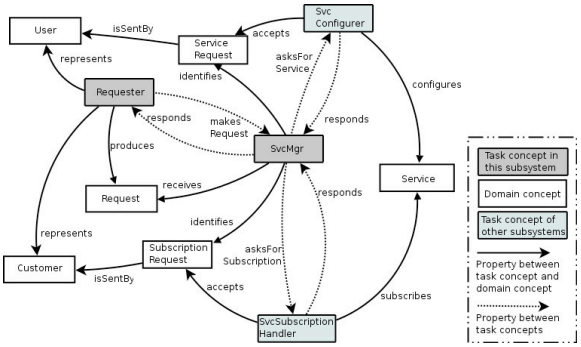

Figure 2. The service management.

The property of *subscribes* represents a series of actions that are then to be elaborated into task ontology of service subscription handling. The *uses* property covers two important properties, *configures* and *observes*, which represent two action series that are then refined respectively as task ontology of service configuration and of service observation.

4.2 Service Subscription

The Service Subscription handles service subscription request from customers and updates SKC every time a subscription is made. The task ontology of service subscription is shown in Figure 3.

4.3 Service Configuration

The Service Configuration responds to a user's request of service. It verifies the request against related license agreements from the SKC before bringing the service into the ready-to-serve state. This service configuration task ontology was described in detail in [2], including the nine roles of the subsytem, which will not be repeated here.

4.4 Service Observation

The Service Observation monitors, analyses, and controls a service, from the usage data and according to the related license agreement from SKC, preventing service abuse and generating usage reports. This task ontology is shown in Figure 4. This task requires that actions are taken by the monitoring system:

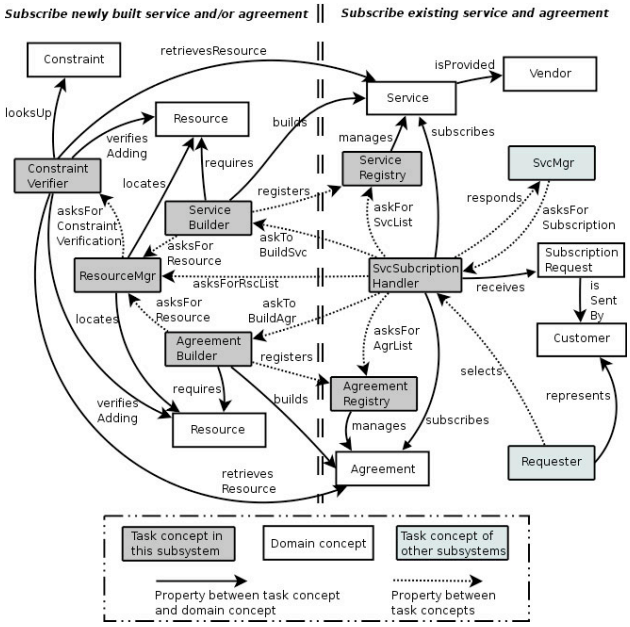

Figure 3. The service subscription.

- When the configuration subsystem notifies that a service needs to be monitored, the observation subsystem locates the agreement and prepares for observation, and depending on the agreement terms decides on which metrics are to be monitored.
- The subsystem analyses and compares these data with the agreement to ensure compliance (ie dynamic verification).
- The subsystem can issue actions to change the status of the service.
- When a service closes, the subsystem reports to the configuration subsystem.

Several task ontology concepts with Service Observation can be implemented as agents that read and write within the various domain concepts, which are accessed for monitoring (the Service for example), parsing (the license terms of the agreement in the KBC), or even control.

5 An Example Ontology for a License

At the heart of any license management system is a representation of the License Agreement (LA) that itemises the consensus of terms between the vendor and customer, terms that describe the duties and constraints of both parties. An example would be the 'time-based license' where one of the

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

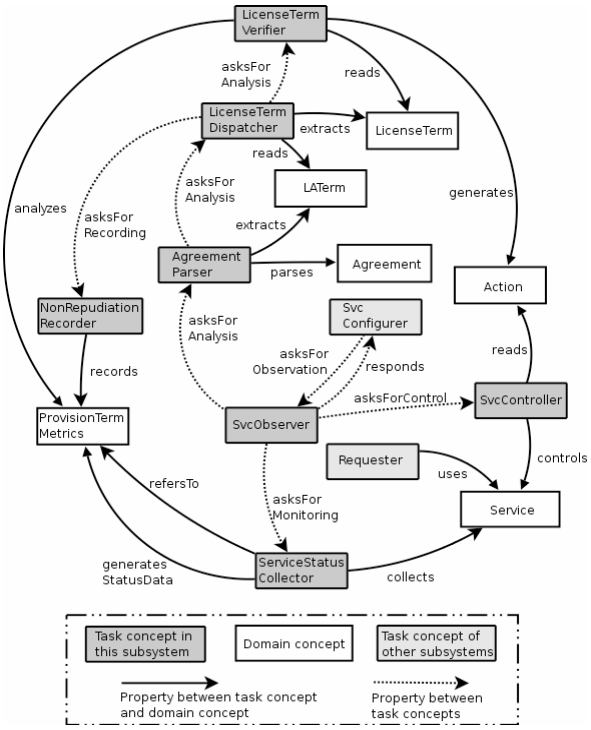

Figure 4. The service observation.

terms is that the use of the service is only available for a certain time; usually with a cut-off date, it is permissive in that it allows the customer to use the software, but constraining in that it can only be used until the given date. Another example would be a 'consumptive license' that allows for a defined number of uses of the service. Every license contains terms with their individual metrics that quantify and qualify an aspect of the service. The domain ontology for a License Agreement is illustrated in Figure 5.

As a simple prototype we take the example of a company (A Co) that provides a service (Service B) to a customer (C). The provision in the license is that the term is for one year, and during the subscription period only 500 concurrent users are permitted, and during the subscription period there can be a maximum of 500,000 transactions. If any constraint is exceeded, then the system must consult the policy for customer C that determines what action is to be taken. The conditions lead to the creation of a group of ontology tuples. Instances of domain and task concepts are created and related to each other. The example gives these statements:

LA001-isA-LA
LA001-hasCustomer-C
LA001-hasVendor-A
LA001-regulates-B
B-isA-Service
B-compliesWith-LA001
LA001-hasProvision-LATerm001
LATerm001-isA-LATerm
LATerm001-hasOperator-AND
LATerm001-contains-LicTerm001
LicTerm001-isA-LicenseTerm
LicTerm001-hasMetrics-Metrics001
Metrics001-isA-TimeMetrics
Metrics001-starts-01Jan2007
Metrics001-ends-31Dec2007
LicTerm001-then-Update001
Update001-isA-Action
Update001-isActedBy-TimeTermVerifier
TimeTermVerifier-isA-LicenseTermVerifier
LATerm001-contains-LATerm002
LATerm002-isA-LATerm
LATerm002-hasOperator-AND
LATerm002-contains-LicTerm002
LicTerm002-isA-LicenseTerm
LicTerm002-hasMetrics-Metrics010
Metrics010-isA-ConcurrentMetrics
Metrics010-hasMax-5,000
Metrics010-hasUnit-NumberOfUsers
NumberofUsers-isA-Unit
LicTerm002-then-Update010
Update010-isA-Action
Update010-isActedBy-ConcurrentTermVerifier
ConcurrentTermVerifier-isA-LicenseTermVerifier
LicTerm001-else-Consult010
Consult010-isA-Action
Consult010-isActedBy-ConcurrentTermVerifier
LATerm002-contains-LicTerm003
LicTerm003-isA-LicenseTerm
LicTerm003-hasMetrics-Metrics003
Metrics010-isA-ConsumptiveMetrics
Metrics010-hasMax-500,000
Metrics010-hasUnit-NumberOfTransactions
NumberofTransactions-isA-Unit
LicTerm003-then-Update003
Update003-isA-Action
Update003-isActedby-ConsumptiveTermVerifier
ConsumptiveTermVerifier-isA-LicenseTermVerifier
LicTerm003-else-Consult003
Consult003-isA-Action
Consult003-isActedBy-ConsumptiveTermVerifier

A simplified view of the set of tuple instances defining the license agreement terms is shown in Figure 6. These domain concepts interact with the task concepts to deal with

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

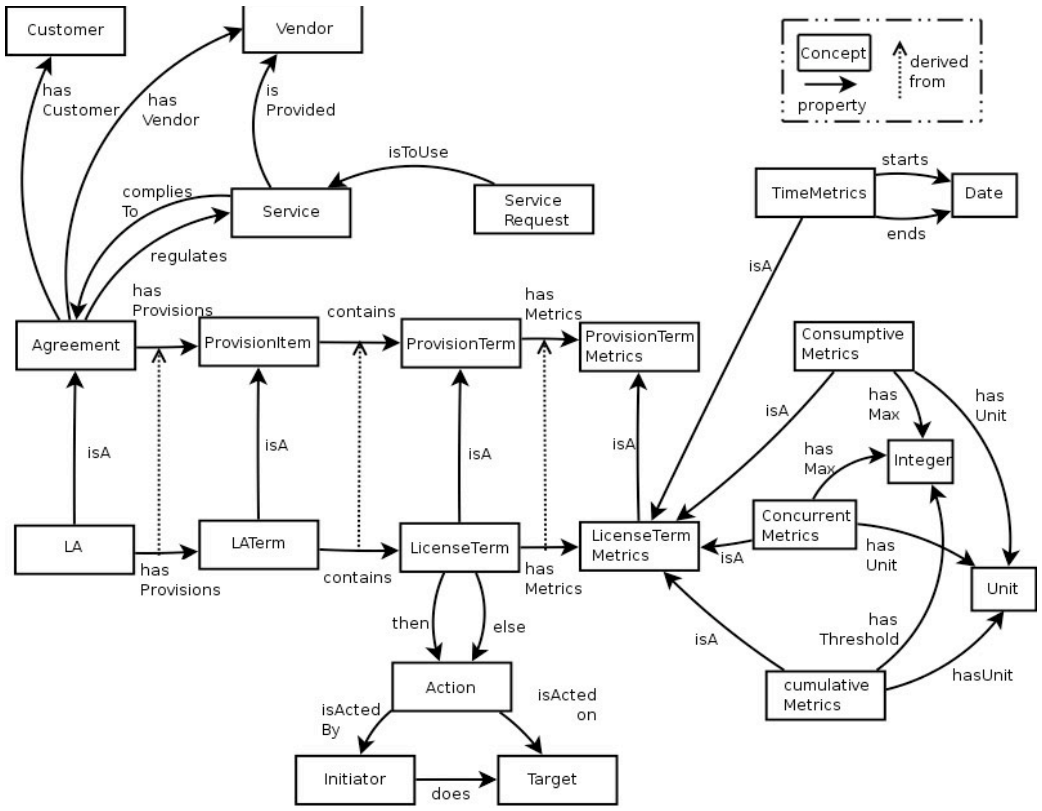

Figure 5. Conceptualising the license agreement.

the license terms in the example agreement before the service B is exposed to C. Figure 7 shows static verification performed by the LicenseTermDispatcher and LicesnseTermVerifiers operate, and in particular how a verifier plays its role accommodate the actions of the license terms.

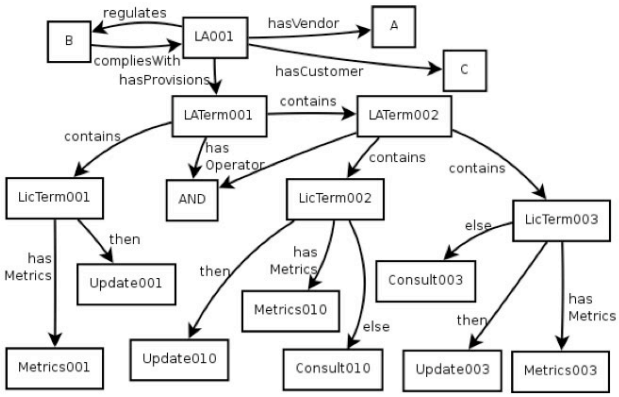

Figure 6. Domain ontology of the example.

6 Conclusion

The joy of a design such as this is that is allows for extension of the system without reinventing the framework. For example, the relation "*a license term regulates a service*" can be extended to "*a policy regulates a service*" where *regulates* is an actionable property in the task ontology that may represent a series of simple operations between concept *policy* and concept *service*. This flexibility depends on the ability to construct an ontology that reflects true world given environments, rather than the simplified prototypes that we have build, but it offers the the potential to control other resources and constraints with the SmArt ontology system. This is work for the future.

Acknowledgment

The authors thank Natural Science and Engineering Research Council and IBM for their support of this research.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.

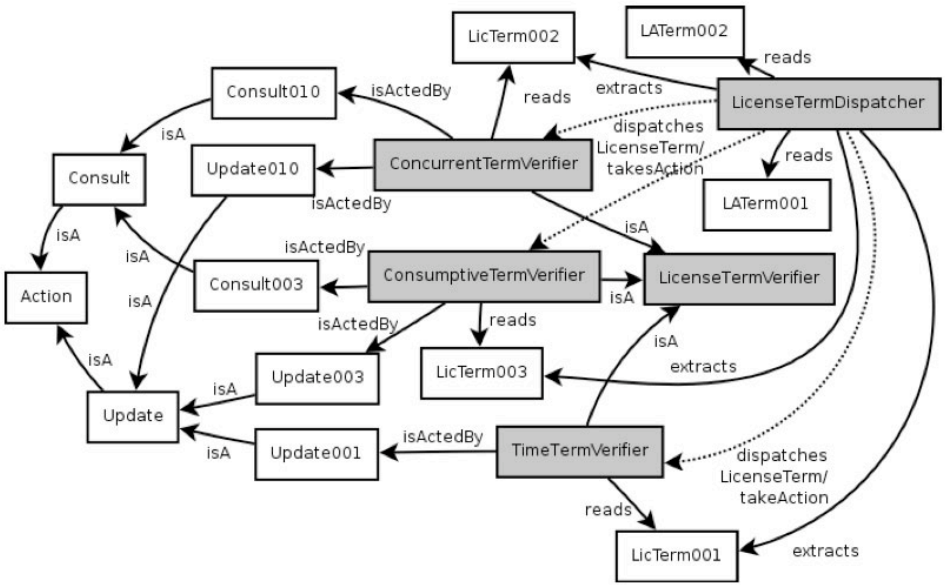

Figure 7. Task ontology of the example.

References

[1] Q. Zhao, Y. Zhou, and M. Perry, "Agreement-aware Semantic Management of Services," in *Proceedings of International Conference on Autonomic and Autonomous Systems*, International Conference on Autonomic and Autonomous Systems 2006. IEEE, 2006.

[2] ——, "Agent design of smart license management system using gaia methodology," in *ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 9.

[3] D. Lenat, M. Prakash, and M. Shepherd, "Cyc: Using Common Sense Knowledge to Overcome Brittleness and Knowledge-Acquisition Bottlenecks," *AI Magazine*, vol. 6, no. 4, pp. 65–85, 1986.

[4] G. O. Consortium, "An Introduction to the Gene Ontology."

[5] C. Rosse and J. Mejino, "A Reference Ontology for Bioinformatics: The Foundational Model of Anatomy," *Journal of Biomedical Informatics*, vol. 36, no. 6, pp. 478–500, 2003.

[6] *The Syntax of CycL*, Cycorp Inc, 2002.

[7] H. Kaminski and M. Perry, "Employing intelligent agents to automate sla creation," *Emerging Web Services Technology*, pp. 33–47, 2007.

[8] Q. Zhao, "Policy-based License Management System for Distributed Software Components," Master's thesis, The university of western ontario, 2003.

[9] H. Kaminski and M. Perry, "Pattern language for software licensing," in *Proceedings of EUROPLoP*. PLoP, July 2005, pp. 177–219.

[10] The-Open-Group, "Technical standard: Systems management: Software license use management (xslm)," XSLM working group, Tech. Rep., March 1999. [Online]. Available: http://www.opengroup.org/onlinepubs/009619399/toc.pdf

[11] WebOpedia, "What Is Service Level Agreement."

[12] E. Wustenhoff, "Service Level Agreement in the Data Center," Sun Microsystems Inc., Tech. Rep., 2002.

Authorized licensed use limited to: UNIVERSITY OF WESTERN ONTARIO. Downloaded on October 28, 2008 at 14:21 from IEEE Xplore. Restrictions apply.