Semantic Modeling for Cloud Computing, Part 2

Amit Sheth and Ajith Ranabahu • Wright State University

Part 1 of this two-part article discussed challenges related to cloud computing, cloud interoperability, and multidimensional analysis of cloud-modeling requirements (see the May/June issue). Here, we look more specifically at areas in which semantic models can support cloud computing.

Opportunities for Semantic Models in Cloud Computing

Semantic models are helpful in three aspects of cloud computing.

The first is functional and nonfunctional definitions. The ability to define application functionality and quality-of-service details in a platform-agnostic manner can immensely benefit the cloud community. This is particularly important for porting application code horizontally — that is, across silos. Lightweight semantics, which we describe in detail later, are particularly applicable.

The second aspect is data modeling. A crucial difficulty developers face is porting data horizontally across clouds. For example, moving data from a schema-less data store (such as Google Bigtable) to a schema-driven data store such as a relational database presents a significant challenge. For a good overview of this concern, see the discussion of customer scenarios in the "Cloud Computing Use Cases White Paper" (www.scribd.com/doc/18172802/Cloud-Computing-Use-Cases-Whitepaper). The root of this difficulty is the lack of a platform-agnostic data model. Semantic modeling of data to provide a platform-independent data representation would be a major advantage in the cloud space.

The third aspect is service description enhancement. Clouds expose their operations via Web services, but these service interfaces differ between vendors. The operations' semantics, however, are similar. Metadata added through annotations pointing to generic operational models would play a key role in consolidating these APIs and enable interoperability among the heterogeneous cloud environments.

Functional Portability

From a perspective of the cloud landscape based on the language abstraction and type of semantics (that is, viewing the cube in Figure 1 from the top), we see that opportunities exist to use semantic models to define applications' functional aspects in a platform-agnostic manner. In most cases, however, converting a high-level model directly to executable artifacts pollutes both representations. Intermediate representations are important to provide a convenient conversion.

Applying high-level modeling to describe an application's functional aspects isn't new. Many software development companies have been using UML to model application functionality at a high level and use artifacts derived from these models to drive development. This process is commonly called model-driven development. This is an example of using high-level models to derive fine-grain artifacts. UML models usually don't include code, so you can use them only to generate a skeletal application. That is, low-level details are deliberately kept away from the high-level models.

UML models, however, are inherently bound with object-oriented languages, and UML-driven development processes depend heavily on advanced tools (for example, IBM's Rational Rose; www-01.ibm.com/software/awdtools/developer/rose). This limits UML's applicability.

A popular alternative to such tool-dependent heavy upfront models is domain-specific languages (DSLs). Their popularity is due partly
Figure 1. A multidimensional slicing of the cloud computing modeling space. This slicing places existing modeling efforts in their rightful part of the space and demonstrates which portions of the modeling space haven’t yet been addressed.

Figure 2. Annotating (a) a domain-specific language (DSL) script with (b) high-level semantic models. These annotations enable the use of rich modeling when required but maintain a simple, convenient representation.

Annotations linking the DSL script to high-level models

(a) configuration do
  config :cost do
    param :max, :value => 1, :unit => :dollar, :period => :hour, 
    :ref => "sla:cost"
  end
  config :perf do
    param :max_response, :value => 20, :unit => :seconds
  end
end
(b)

Data Functional Nonfunctional System

between executable artifacts and high-level semantic models. A DSL, although domain specific, can provide a more programmer-oriented representation of functional, non-functional, or even data descriptions.

A best-of-both-worlds approach is to use annotations to link models, which provides the convenience of lightweight models while supporting high-level operations when required. Figure 2 shows an annotation referring to an ontology from a fictitious DSL script for configuration. The script is more programmer-oriented (in fact, it’s derived from Ruby) but lacks an ontology’s richness. However, the annotation links the relevant components between the different levels, providing a way to facilitate high-level operations while maintaining a simpler representation.

From the perspective based on the type of semantics and software life-cycle stage — that is, looking at the cube in Figure 1 from the front — you can see the modeling coverage for software deployment and management. Elastra’s Elastic Computing Modeling Language (ECML), Elastic Deployment Modeling Language (EDML), and Elastic Management Modeling Language (EMML) ontologies cover many system aspects and some non-functional aspects in all stages. We’re pleased to see such industry initiative in adopting semantic technologies.

Data Modeling

Another opportunity for semantic models for clouds lies in RDF data modeling. As we discussed in part 1, a major concern plaguing cloud computing’s adoption is data lock-in — that is, the inability to port data horizontally. Many vendors designed schema-less, distributed data stores with relaxed consistency models to provide high availability and elasticity to suit clouds’ needs. However, exploiting these data stores requires substantial redesign of many data-driven

to the availability of extensible interpreted programming languages such as Ruby and Python. Unlike UML, a DSL is applicable only in a given domain but enables a lightweight model in that domain, often without requiring proprietary tools. For example, you can use IBM’s Sharable Code DSL (http://services.alphaworks.ibm.com/isic), which is a mashup generator, with a basic text editor. (However, providing graphical abstractions and specialized tooling would be more convenient for users.) "Lightweight" signifies that these models don’t use rich knowledge representation languages and so have limited reasoning capabilities.

Our CirroCumulus project for cloud interoperability (http://knoesis.org/research/srl/projects/cirrocumulus) uses DSLs to bridge the gap
Applications and often makes porting data to a traditional relational database extremely difficult. The current practice is to address such transitions case-by-case.

A better approach is to model the data in RDF and generate the specific target representations and, in some cases, even the code for the application's data access layer. This method can formulate transformations from one representation to another using the lifting-lowering mechanism. Semantic Annotations for WSDL and XML Schema (SAWSDL; www.w3.org/TR/sawSDL) demonstrated this mechanism's use for data mediation. Lightweight modeling in terms of DSLs also applies here. For example, the Web services community has long used XML Schema definitions as platform-agnostic data definitions. Schema definitions serve as inputs to code-generation tools that generate platform-specific data definitions.

From the perspective of the type of semantics and software life-cycle stage, most of this data modeling applies during application development. Concrete artifacts generated from these high-level models would be used mostly during subsequent life-cycle stages.

**Service Enrichment**

One feature differentiating the cloud from other distributed environments is the availability of Web services to manipulate resources. Availability of the service APIs lets you programmatically manage the cloud resources, even from within the same cloud. These capabilities have revolutionized application deployment and management. For example, you can compose or mash up cloud services to facilitate elaborate deployment workflows.

Service definitions are usually syntactic, and many researchers have focused on embedding rich metadata in formal service descriptions. One result of this research is SAWSDL. A growing trend is to annotate HTML descriptions to embed richer, machine-readable semantic metadata. One reason for this method's popularity is search engines' use of metadata to display results in customized formats. Yahoo's SearchMonkey and Google Rich Snippets are two such microformat-driven schemes. These annotations, unlike the DSL annotations in Figure 3, might not always point to ontologies. Their structure can be based on a vocabulary or taxonomy—a lower-grade nonsemantic model. For example, the popular hCalendar microformat is part of the "lowercase semantic web" movement, which emphasizes lightweight models.

Embedding rich semantic metadata in cloud service descriptions has three main benefits that go beyond customized search capabilities.

* The first benefit deals with Representational State Transfer (REST) style services. Many cloud service providers adopt REST-style Web services that don't advocate a formal service description. These services are described using HTML pages. WSDL 2.0, the latest specification, explicitly supports formal description of RESTful services but hasn't seen much adoption. Alternative approaches such as SA-REST (SA stands for semantic annotation), a generic annotation scheme that follows microformat design principles, are becoming more applicable in this space. These annotations enable the seamless, flexible integration of formalizations into RESTful service descriptions. This opens the door to many exciting avenues such as faceted search to identify relevant reusable services and semiautomated service compositions.
The second benefit deals with handling change. The cloud space is still evolving. If the history of software or component interoperability is any guide, achieving consensus in the cloud space will be difficult and won’t likely happen soon. Attaching formalizations via annotations, however, is flexible enough to accommodate an evolving model. This is especially attractive to vendors who aren’t willing to invest heavily in interim standards.

The third benefit is that the formalizations apply not only to service descriptions but also to many other aspects such as service-level agreements (SLAs) and software licenses. You can use annotations to embed formalizations even for these documents, facilitating more automation in the cloud space. For example, the Web Service-Level Agreement (WSLA; www.research.ibm.com/wsla) specification provides a way to formalize SLAs, but creating and maintaining these formalizations is time-consuming.

Figure 3 illustrates using SA-REST annotations on the Amazon Elastic Compute Cloud (EC2) SLA document. It shows how a capable processor could use these annotations to extract a WSLA equivalent of the human-readable SLA.

These benefits’ importance comes into perspective when you consider the enormous body of research on standard-driven service compositions and agreement matching. The informal, non-standard-driven nature of many cloud services made most of the previous research inapplicable. However, being able to glean formalizations from existing documents opens the doors to apply many well-researched techniques.

The cloud space presents many opportunities for researchers, and we see a plethora of applications that use semantic modeling.

Issues such as interoperability and data portability, which the cloud community is facing right now, are the very issues for which semantic models excel in providing solutions. However, learning from the past, we advocate a multilevel modeling strategy to provide smooth transitions among different granularities. We also think that DSLs can play an important role in the cloud space to provide lightweight modeling in an appealing manner to the software engineering community. [ib]

References

Amit Sheth is the director of the Ohio Center of Excellence on Knowledge-Enabled Computing (Kno.e.cs) at Wright State University. He’s also the university’s LexisNexis Ohio Eminent Scholar. He’s on the Web at http://kno.e.cs./amit.

Ajith Ranabahu is pursuing a PhD in cloud computing interoperability at Wright State University. He worked with IBM on Sharable Code and its Attocumulus project, and he coordinates the Cirrocumulus project (http://kno.e.cs./wright.edu/research/sri/projects/cirrocumulus). Contact him at ajith@kno.e.cs.org.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

IEEE COMPUTER SOCIETY

Purpose: The IEEE Computer Society is the world’s largest association of computing professionals and is the leading provider of technical information in the field. Visit our Web site at www.computer.org.

Ombudsman: Email help@computer.org.

Next Executive Board Meeting: 15–16 Nov. 2010, New Brunswick, NJ, USA

Executive Committee
President: James D. Isaac

*Voting member of the Board of Governors

Honorary members of the Board of Governors are James C. Henderson and John Kemeny.

Board of Governors
Tom Epling 2010: Flore Bourgeois; André Ivanov; Philip A. Laplante; Jarke Miyata; Jon G. Rokne; Christina M. Schauer; Ann E.K. Sobel
Term Epling 2011: Eliza Berlino, George V. Cybenko, Ann DeMaria, David S. Ebert, David A. Grier, Hiroshi Iwashina, Steven L. Tanimoto
Term Epling 2012: Elizabeth L. Bird, Thomas M. Conti, Frank E. Ferrante, Joan-Luc Gaulet, Luis Kun, James W. Moore, John W. Walz

Executive Staff
Executive Director: Angela R. Burgess; Associate Executive Director: Director, Governance; Anne Marie Kelly; Director, Finance & Accounting: John Miller; Director, Information Technology & Services: Carl Scat; Director, Membership Development: Violet D. Doan; Director, Products & Services: Evan Butterfield; Director, Sales & Marketing: Dick Price

Computer Society Offices
Phone: 1 202 371 0100; Fax: 1 202 728 9514; Email: help@computer.org
Los Alamitos: 10836 Los Vaqueros Circle, Los Alamitos, CA 90720-1314
Phone: 1 714 821 8300; Email: help@computer.org
Membership & Publications
Phone: 1 800 272 6657; Fax: 1 714 821 6441; Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • Fax: +81 3 3408 3553
Email: tokyo.ofc@computer.org

IEEE Officers
President: Pedro A. Ray; President-Elect: Moshe Kam; Past President: John B. Vig; Secretary: David G. Grier; Treasurer: Peter W. Staeler; President, Standards Association Board of Governors: W. Charlotte Adams; VP Educational Activities: Tony J. Durrani; VP Membership & Geographic Activities: Barry L. Smith; VP Publication Services & Products: Jon G. Iwakawa; VP Technical Activities: Roger D. Pollard; IEEE Division V Director: Michael R. Williams; IEEE Division VIII Director: Stephen L. Diamond; President, IEEE-USA: Evelyn H. Hirt

IEEE INTERNET COMPUTING