Editors: Michael Rabinovich • *mis@brics.dk*
Steve Vinoski • *vinoski@ieee.org*

Active Web Service Registries

Atom news feeds can work both as an infrastructure for distributed Web service registries and to inform users about Web service changes. The authors consider an approach in which every Web service provider offers a Web service registry news channel that serves as part of a globally distributed registry. They also apply this approach to a real-world case study. Their future plans include providing a Web service aggregator that offers additional metadata for Web service descriptions, which users could generate as they rate the services they use.

Martin Treiber and Schahram Dustdar
Vienna University of Technology

Web service registry implementations such as UDDI and Electronic Business XML (ebXML) registries have had limited success for several reasons, including their centralized architectures, their entries' limited accuracy, complicated registration and removal processes, a lack of suitable browsers for users, and complex APIs for programmers.[1]

Although a centralized architecture offers a place to discover Web services, this architectural decision raises additional problems, including scalability issues and a single point of failure. Major companies such as IBM and Microsoft, for example, have recently disconnected their public (centralized) registries (http://uddi.microsoft.com/about/FAQshutdown.htm), forcing Web service requesters to get Web service-related information via email. Clearly, this situation isn't satisfactory and requires a new, standards-compliant mechanism.

Following related research (www.ipbabble.com/2006/01/rss_demo_code.html), we offer an approach in which an RSS-based infrastructure overcomes these problems with current registry tactics (see the "Related Work in Web Service Registries" sidebar for additional research). RSS, in which a simple XML-based document holds either a summary of content from an associated Web site or the full text, has become an extremely popular

1089-7801/07/$25.00 © 2007 IEEE

Related Work in Web Service Registries

Several researchers in the Web services community have already proposed various distributed Web service registry approaches. Some even use distributed hash tables[1,2] to distribute content.[3] In our approach, however, we don't use a hash table function in this manner, thus we don't reorganize registry content on the peers in a structured overlay network. Instead, we build an unstructured network of active Web service registry (AWSR) feeds, which allows for a tight coupling of Web service registry content to the Web service provider and provides accuracy by keeping registry information at the provider.

The Web service registry federation approach from Michael P. Papazoglou and colleagues[4] introduced the concept of a UDDI-based Web service registry federation. The authors rely on the concept of a *super peer* — a dedicated peer in the peer network that manages syndication and provides a local syndication UDDI registry (a subset of a global UDDI registry). Our approach differs slightly: we don't use UDDI registries (instead, we favor a lightweight approach based on RSS 2.0). We also provide a fully distributed Web service registry solution that supports a hierarchical structure of Web service registries, including their taxonomy information.

Other registry federation approaches[5,6] focus on the integration of heterogeneous Web service registries. In contrast, we use a mashup approach, in which the existing RSS infrastructure is reused to build an AWSR. XMethods (www.xmethods.net) is somewhat similar — its news feed informs subscribers about recently registered Web services — but it provides only the last 10 published Web services and no means to create a syndication of different Web service news feeds.

Finally, Bahman Kalali and colleagues present a Web service registry that notifies subscribers about changes, but we use Atom news feed mechanisms to provide notifications about changes in a fully distributed Web service registry.

References

1. I. Stoica et al., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. SIGCOMM*, ACM Press, 2001, pp. 149–160.
2. A.I.T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Middleware*, Springer-Verlag, 2001, pp. 329–350.
3. C. Schmidt and M. Parashar, "A Peer-to-Peer Approach to Web Service Discovery," *World Wide Web*, vol. 7, no. 2, 2004, pp. 211–229.
4. M.P. Papazoglou, B.J. Kramer, and J. Yang, "Leveraging Web-Services and Peer-to-Peer Networks," *CAiSE*, Springer-Verlag, 2003, pp. 485–501.
5. T. Pilioura, G.-D. Kapos, and A. Tsalgatidou, "Seamless Federation of Heterogeneous Service Registries," *EC-Web*, Springer-Verlag, 2004, pp. 86–95.
6. M. Treiber and S. Dustdar, "Integration of Transient Web Services into a Virtual Peer to Peer Web Service Registry," *Distributed and Parallel Databases*, vol. 20, no. 2, 2006, pp. 91–115.
7. B. Kalali, P.S.C. Alencar, and D.D. Cowan, "A Service-Oriented Monitoring Registry," *Proc. 2003 Conf. Centre for Advanced Studies on Collaborative Research*, IBM Press, 2003, pp. 107–121.

way to distribute frequently updated content to subscribers. The tooling support for RSS has also grown considerably, with numerous RSS news reader implementations now available, and several email clients now supporting RSS feeds. So-called news feed aggregators filter and assemble content from different RSS feeds into a single, coherent RSS feed (www.newsonfeeds.com/faq/aggregators). Different tools provide management facilities for creating the feeds (see www.jitbit.com/rssfeedcreator.aspx and www.feedforall.com/feedforall.htm).

In our approach, we use available RSS software to construct an active distributed Web service registry. We also extend the RSS data model with Web service-specific information, such as links to Web Services Description Language (WSDL) files.

A Real-World Scenario

To motivate our work, we focused on the Web service requirements of an Austrian firm – Wirtschaftsauskunftei Wisur (www.wisur.at) – that provides business reports and other business-related company information (www.wisur.at/pdf/bus_en.pdf).

To help potential customers find the company's Web services, Wisur registered itself in a public UDDI Web service registry, but that registry shut down in January 2006. The company now has two choices: find another public UDDI Web service registry or set up one on its own. The problem with the first option is that other public registries might not exist indefinitely, so the more plausible alternative is the second choice. Unfortunately, setting up a registry involves considerable overhead, including database management, user management, installation of registry browsers, and so forth, which is a rather heavyweight approach compared to the small number of Web services provided.

Another issue concerns UDDI registry usage. From a developer's perspective, it's difficult to find relevant technical information about a Web service because the developer must navigate through several different data structures to find it. A better solution would offer the ability to sift through registry content with existing tools, such as Web browsers. Examples of correct Web service invocations should also be available in a Web service registry, to help developers test their implementa-

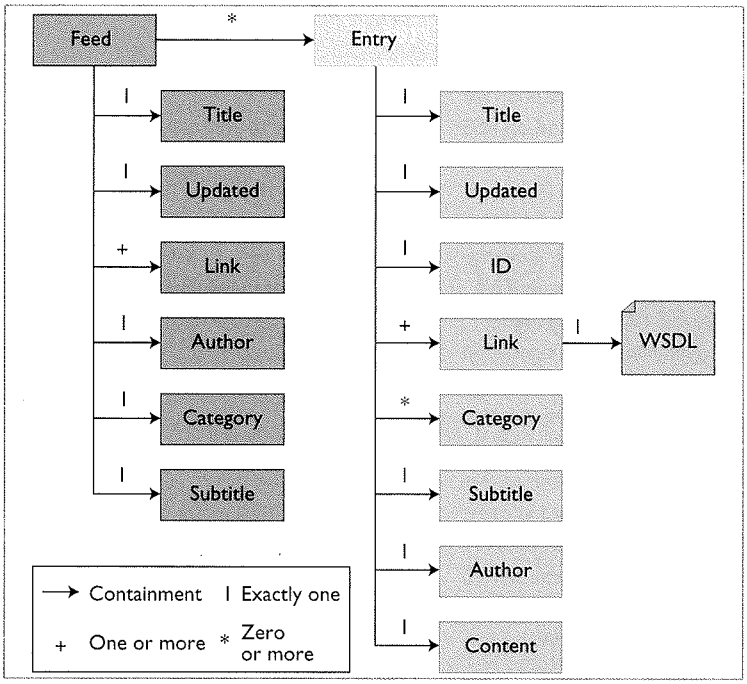


Figure 1. Web service registry data model. We focus here only on what the Web service registry uses.

tions without having the additional communication overhead associated with a Web service provider. Another issue involves changes to existing Web services. Wisur wants to be able to notify business partners about changes to its Web services – for instance, when a new version of a Web service with additional company information becomes available – but change notification currently involves sending email messages to all business partners manually. A Web service registry should support this task instead.

Wisur also cooperates with other companies, offering its partners' services transparently to its customers through a joined registry. However, syndication isn't currently possible, so Wisur and its partner companies depend on their customers knowing where to find the corresponding services. To motivate business partners to participate in Web service registry syndication, Wisur needs an easy-to-use and simple-to-maintain syndication mechanism that doesn't involve too much administrative overhead.

The Web service registry requirements that arise from this real-word scenario are as follows:

- *Reuse of existing infrastructure.* The existing Web server should be reused for the Web service registry.
- *A lightweight approach.* In contrast to existing Web service registry solutions, the new registry shouldn't require complex software installations.
- *A lightweight data model.* The data model should be compatible with an existing RSS data model or provide transparent, lightweight extensions to it, thus allowing the use of standard software such as Web browsers, news readers, and so forth.
- *Pre- and postcondition examples.* Concrete examples of valid Web service invocations should be explicitly part of the Web service registry, thus making it easier for developers to test Web services.
- *Content federation.* The Web service registry should provide a distributed architecture and support the federation of content.
- *User notification.* Registered users should immediately receive changes to Web service registry content.
- *Syndication.* It should be easy to syndicate content from different Web service registries.
- *Content accuracy.* The Web service registry's content shouldn't contain invalid entries and must be tightly coupled with the service provider.
- *Content control.* The Web service provider should directly host the Web service registry.

For Wisur, these requirements led to the development of a lightweight Web service registry.

Active Web Service Registries

To address Wisur's needs, we developed an active Web service registry (AWSR) based on RSS feeds that also contains Web service-related information (such as links to interface descriptions).

Embedding Data in RSS Feeds

In our approach, we use the Atom news format as a vehicle for Web service registry content. We embed registry content directly in Atom's data model, which is organized as a hierarchy of feed elements that hold an arbitrary number of entry elements. We use each feed element as a container for meta information about the Web service registry. Figure 1 provides an overview of Atom's data model. Our approach explicitly requires the presence of some elements that are optional in this model – in particular, we require the presence of at least one category element in the feed that provides for domain-related information. In the Atom newsfeed's entry elements, we require the existence of a link element that points to an external

WSDL file. We also require the presence of at least one category element and a content element that contains an invocation example of the corresponding Web service.

The use of channels and feeds is a very flexible way to structure Web service registry content, making it possible to create customer registry feeds (such as a dedicated feed for those with access to premium services) or Web service channels for internal company use only.

The following code snippet shows Wisur's public Web service registry channel, with all its publicly available services. As shown in the code snippet, we use Atom elements to represent general information about the channel:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/
   Atom">
<title>Wisur Business Services</title>
<subtitle>Wisur Business Services
   provide informations about business
   partners. These services include
   information about company turnover,
   balance sheet data, scoring,
   etc.</subtitle>
<link href="http://www.wisur.at/rss"/>
<updated>2007-05-30T18:30:02Z</updated>
<author>
<name>Wirtschaftsauskunftei Wisur
   GmbH</name>
<email>contact@wisur.at</email>
</author>
<id>urn:uuid:60a76c80-d399-11d9-b93C-
   0003939e0af6</id>
<category scheme='"http://www.dmoz.org'
   term='"http://www.dmoz.org/Business/
   FinancialServices/" />
<entry>
...
</entry>
</feed>
```

As mentioned previously, Atom's feed element contains an arbitrary number of entry elements, and we use these elements to contain Web service descriptions. Every entry element represents a single Web service:

```
<entry>
<title>Wisur Credit Check
   Service</title>
<link href=http://www.wisur.at:8000/
   axis/services/WISIRISSearchService?
   wsdl"/>
<id>urn:uuid:1225c695-cfb8-4ebb-aaaa-
   80da344efa6a</id>
<updated>2007-05-23T20:34:02Z</updated>
<summary>This Web service is the search
   interface to the Wisur Company
   Database. For the use of the Web
   service, a valid User ID is neces-
   sary. This ID can be obtained with
   the registration Service.</summary>
<category scheme='"http://www.dmoz.org'
   term='"http://www.dmoz.org/Business/
   FinancialServices/"/>
<content>
<Search>
<CustomerID>12402</CustomerID>
<Name>Bauer</Name>
<Key>32559fae-5448-47f4-bea6-fecf386b
   e580</Key>
</Search>
</content>
</entry>
```

We embed general, human-readable information in both the title and summary elements; domain-related information is included in the category element, and the link element references an external WSDL description of the Web service. By following the link element, users can access the Web service's interface description directly, without having to traverse several data structures before finding it. This is in contrast to other data models, such as UDDI, where technical information is represented by `tModels` nested in other data structures.

In addition, we use the content element to provide for additional information about the corresponding Web service. In our current implementation, we included best practices that show by example how the corresponding Web service is called. In other words, we provide a concrete SOAP message that developers can use to invoke the Web service. In this way, it's possible to test a Web service without having to contact the Web service provider and ask for test data.

Publication and Discovery of Web Services

To offer Web services to potential business partners, a Web service provider must publish those services in a common Web service registry, but rather than require each provider to use a cen-

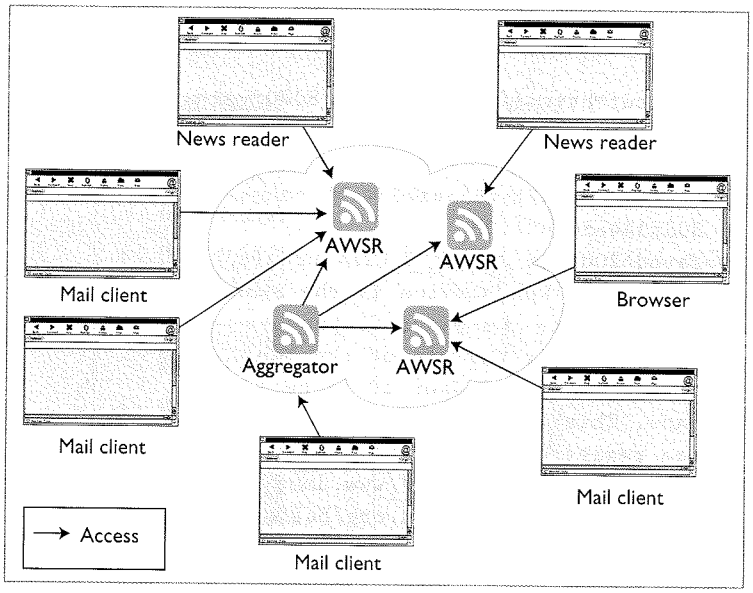


Figure 2. Client- and server-side active Web service registry (AWSR) aggregation. The aggregator unifies three AWSR feeds and then provides a single feed back to a single client; the other clients connect directly to AWSR registries.

tralized place, we follow a decentralized approach. In our concept, every Web service provider offers an AWSR registry feed simultaneously, tightly coupling registry content to providers and maximizing that content's accuracy. This approach also leverages the provider/requestor/registry triangle toward a peer-to-peer concept, with provider and consumer independent from a third authority.

The publication process itself is straightforward: it just consists of writing a new feed item into the AWSR feed. The discovery process consists of subscribing to an AWSR with existing tools. Depending on the tool's capabilities, users can choose different filter options to personalize AWSR content; RSS feed notification mechanisms propagate any changes. All subscribed users get the new Web service descriptions as soon as they poll their AWSR feeds.

Syndication of Web Service Registries in RSS Feeds

We can use RSS syndication mechanisms to fuse distributed AWSR feeds. In our approach, we use the hierarchical categorization scheme from the Open Directory Project (www.dmoz.org). Two key concepts in our syndication model are the notions of client- and server-based Web service registry syndication (see Figure 2). The former syndicates Web service content from different sources – that is, the client registers to several AWSR feeds and generates a local virtual Web service registry. The latter syndicates AWSR feeds from different sources and provides a single, coherent feed. In contrast to client-side syndication, subscribers are unaware that the AWSR feed consists of several single AWSR feeds.

Discovery of Active Web Service Registries

Our approach provides two different ways for discovering distributed Web service registries. The first uses the RSS feeds' autodiscovery feature, which consists of adding a link tag to a Web page's header and letting the user decide whether to subscribe to the registry. The second way relies on index sites, which are similar to RSS news aggregator sites. Interested parties publish their AWSR feeds at a well-known location, thereby providing a single point for AWSR feed discovery. A prototype implementation is available at www.vitalab.tuwien.ac.at/projects/visco/awsr/; it supports the registration of AWSRs and provides basic search capabilities for registered AWSR feeds.

Application Scenarios

Let's review some application scenarios for AWSRs.

Pushing Services to Customers

In this scenario, Wisur actively contacts other companies to conduct business. The whole business process consists of several steps:

1. A Wisur salesperson contacts companies potentially interested in Wisur's services.
2. If a company is interested in these services, then the salesperson negotiates a contract (defining payment modalities, discounts, and so forth).
3. After the contract is completed, either Wisur's developers contact the business partner's developers or vice versa, usually over email or by telephone.
4. Wisur's developers provide information about using test services, including details about the protocols used and the endpoints for Web service invocation. To support the development, Wisur's developers also present test cases to the partner's developers.
5. After the test phase, the partner's developers obtain production information (valid user IDs, user keys, and so on) from Wisur and "switch" to production services.

6. When the contract terminates, Wisur removes the user IDs and keys, and the corresponding Web services can no longer be invoked.

The application of AWSR in this scenario occurs at the developer level, during steps 3 and 4: the business partner's developers register with Wisur's AWSR news feed and then browse all the necessary Web service information to integrate Wisur's Web services into their system. In this scenario, AWSR's main benefit is that the developers don't need specialized software to get registry content. Moreover, the data model is considerably simpler and provides direct access to relevant information.

Change of Business Services

In this scenario, established services change due to requirement changes (for example, a business partner needs an extension to an existing service). In this case, the process consists of the following steps:

1. Wisur's developers contact the business partner's developers, usually over email or by telephone.
2. The partner's developers provide information about the desired Web service changes. Wisur implements the new version of the Web service and provides test cases to ensure that it addresses all the new requirements. The final steps (test and production phases) are the same as in the previous section.

The application of AWSR in this scenario starts with step 1: as soon as a Web service changes, Wisur's AWSR feed informs developers about any state changes. As soon as a new version of a Web service is available and is published in the AWSR registry, all registered customers know about the new service offerings.

Web Service Registry Syndication

In this scenario, Wisur cooperates with a partner company whose set of Web services should be offered Wisur's customers too. Because of marketing issues, the partner company is interested in Web service offerings under Wisur's "umbrella."

The application of AWSR in this scenario involves the syndication of several different Web service registries. Wisur registers the partner's AWSR feed and provides a single AWSR feed for all its customers. To provide a coherent feed, business partners must create an AWSR-compliant Atom news feed, usually by creating an XML file that's available on the partner company's Web site. The registration process consists of accessing Wisur's Web site and providing the link to the XML file. The content is automatically integrated into Wisur's registry and becomes accessible for all customers.

We intend to address managed Web service aggregation in the next version of our Web service registry. We plan to provide a Web service aggregator that offers additional metadata for Web service descriptions, which users could generate as they rate the services they use. In this context, E. Michael Maximilien and Munindar P. Singh's work is of interest for further investigations.[2]

We've implemented an active Web service registry feed on Wisur's homepage (www.wisur.at/rss/WISIRISServices.rss). Our current prototype supports the basic Web service registry operations (publish and unpublish), but we don't use Atom's Publishing Protocol. We intend to use it in a future version of our prototype.

Acknowledgments

This work is funded by the FFG (www.ffg.at) as part of the ITEA project Osiris (www.itea-osiris.org).

References

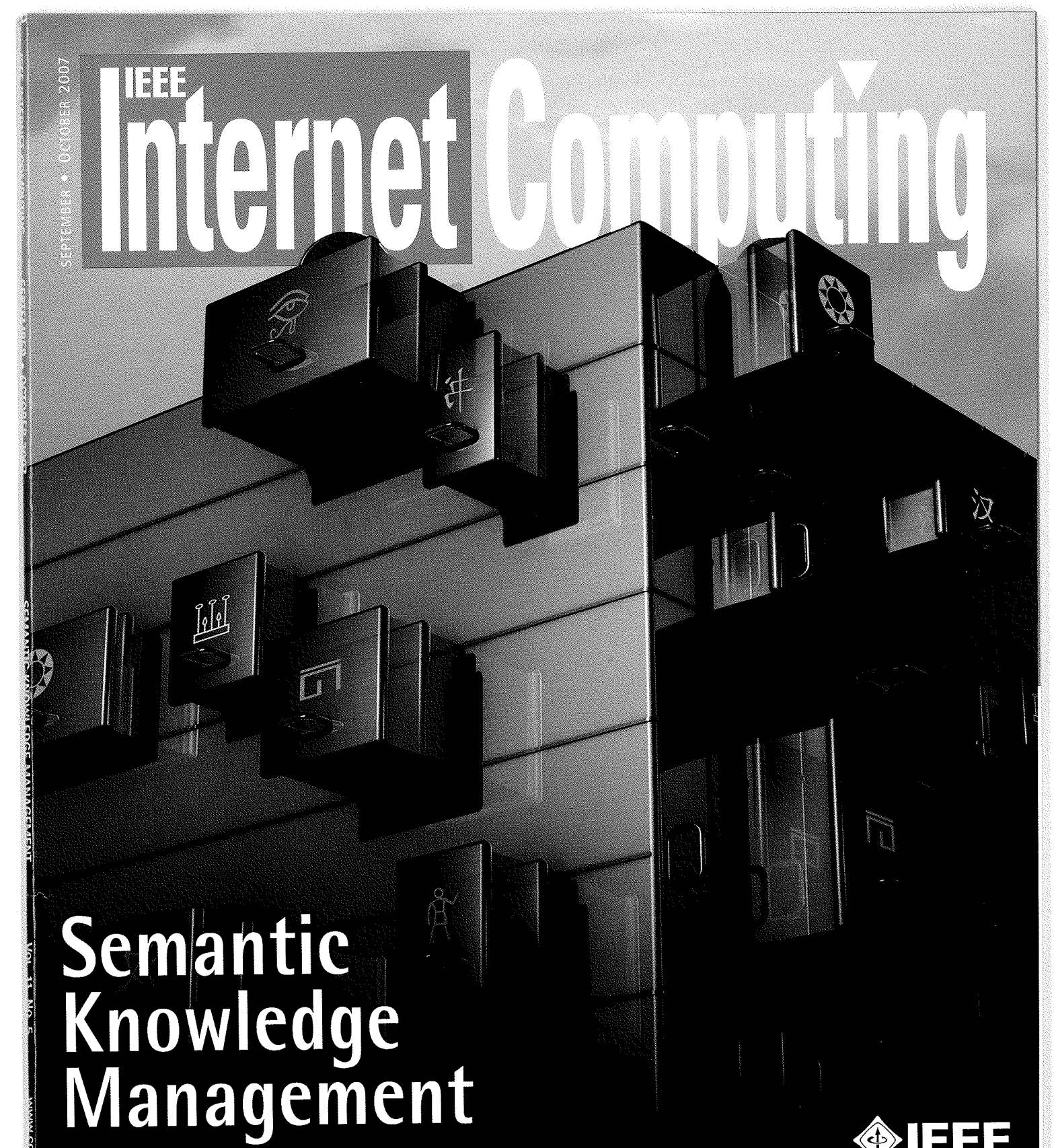
1. M. Treiber and S. Dustdar, "View-Based Integration of Heterogeneous Web Service Registries: The Case of VISR," *WWW J.*, vol. 9, no. 4, 2006, pp. 457–483.
2. E.M. Maximilien and M.P. Singh, "Conceptual Model of Web Service Reputation," *SIGMOD Record*, vol. 31, no. 4, 2002, pp. 36–41.

Martin Treiber is a PhD candidate in the Distributed Systems Group at the Vienna University of Technology. His research interests include service-oriented computing, autonomic computing, and Web-based mashup technologies. Treiber has an MSc in computer science from Vienna Technical University. Contact him at m.treiber@infosys.tuwien.ac.at.

Schahram Dustdar is a full professor in the Distributed Systems Group at the Vienna University of Technology. His research interests include service-oriented architectures and computing, mobile and ubiquitous computing, complex and adaptive systems, and context-aware computing. Dustdar has a PhD in business informatics from the University of Linz. Contact him at dustdar@infosys.tuwien.ac.at.

SEPTEMBER • OCTOBER 2007

IEEE Internet Computing



Semantic Knowledge Management

IEEE computer society

HOME SYSADMINS KEEP BUSY • BEYOND MPLS

SEMANTIC KNOWLEDGE MANAGEMENT

Vol. 11, No. 5

WWW.COMPUTER.ORG/INTERNET/

IEEE Internet Computing

SEPTEMBER/OCTOBER 2007, VOLUME 11, NUMBER 5

DEPARTMENTS

COLUMNS

www.computer.org/internet/

This publication is indexed by ISI (Institute for Scientific Information) in SciSearch, Research Alert, the CompuMath Citation Index, and Current Contents/Engineering, Computing, and Technology.

Circulation: *IEEE Internet Computing* (ISSN 1089-7801) is published bimonthly by the IEEE Computer Society. IEEE headquarters: 3 Park Avenue, 17th Floor, New York, NY 10016-5997. IEEE Computer Society headquarters: 1730 Massachusetts Ave., Washington, DC 20036-1903. IEEE Computer Society Publications Office: 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, Calif. 90720; (714) 821-8380; fax (714) 821-4010.
Subscription rates: IEEE Computer Society members get the lowest rates and choice of media option — US$42/73/130 for print + online/sister society/individual nonmember. For information on other prices or to order, go to www.computer.org/subscribe. Back issues: $25 for members, $130 for nonmembers.
Postmaster: Send undelivered copies and address changes to *IEEE Internet Computing*, IEEE Service Center, 445 Hoes Ln., Piscataway, NJ 08855-1331. Periodicals postage paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8. Printed in the USA.

Printed on 100% recycled paper