Composing Web Services for Large-Scale Tasks

By making it easy to explore combinations of multiple Web services, Eurasia helps users tackle large-scale information-management tasks and adapt and reuse the steps involved.

In-Young Ko and Robert Neches
Information Sciences Institute, University of Southern California

Web services give users access to various information-management services that help them gather, organize, and manage Web-based information. However, it's currently impractical for users to properly manage and execute the many steps required to retrieve, analyze, and visualize information for large-scale tasks, such as performing a multisource intelligence gathering and analysis task or building an automatically updated Web portal. Assuming users do achieve success in such large-scale information, management tasks, it's unlikely that they'd be able to quickly adapt and reuse the many steps involved for other tasks and environments.

Currently, Web services research focuses on developing mechanisms to describe individual services, locate them in a network environment, and access them based on functionalities and constraints. However, to fully realize Web services' potential flexibility and adaptability, we must combine various services to enable large-scale task management. To this end, we propose Eurasia (Exploring, Understanding, and Recording Analysis Steps in Information-Management Applications), a high-level system that helps users quickly explore and test various Web services so that they can compose large-scale applications.

Eurasia evolved from our development of GeoWorlds (www.isi.edu/geoworlds), a Web-based system that makes various network information-management services easily accessible to users.[1] We began the GeoWorlds project in 1998; our work on it provided insight into several key aspects of user behavior in Web-based environments, which in turn inspired our work on Eurasia.

- *Incremental development.* Users' task goals are initially abstract, and they don't know which information-management steps to pursue. By exploring and testing candidate sources and

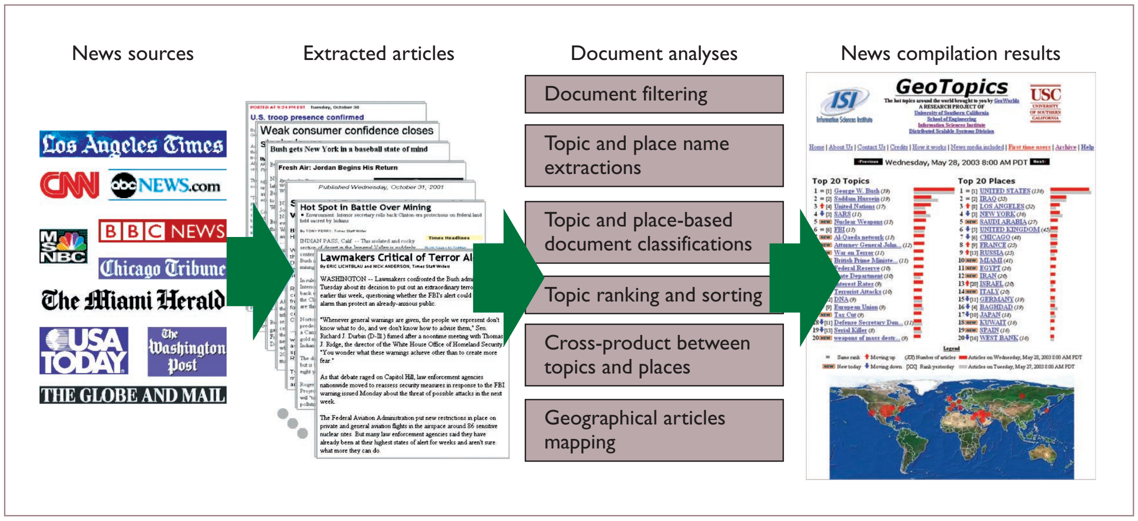

News sources
Extracted articles
Document analyses
News compilation results
Document filtering
Topic and place name extractions
Topic and place-based document classifications
Topic ranking and sorting
Cross-product between topics and places
Geographical articles mapping
GeoTopics

Figure 1. The GeoTopics news analysis application. GeoTopics uses 92 component services to extract daily news from numerous sources, analyze the documents, and present results.

services, users gradually discover which steps are required to accomplish their goals.

- *Recurrent execution.* Users often rerun steps to update their analysis results with the latest Web information.
- *Evolving requirements.* User requirements are continuously changing, and users modify steps based on insights they gain from incoming results.
- *Patterns of collaboration and reuse.* Users exchange similar information-management patterns when they collaborate, and they frequently reuse a similar set of analysis steps for multiple tasks.
- *Use of document collections.* After retrieving various Web documents, users typically reorganize them into different structures and display the collection in different formats.
- *Dynamic information management.* Because the availability of Web sources and services changes over time, users must adjust their information-management steps accordingly.
- *Choice of services.* Users need help sorting through the many available information retrieval, analysis, and visualization services that come in various forms, including software components, Internet agents, and Web services.

Eurasia is currently in experimental use in several public and military domains. Application developers have used Eurasia to build more than 45 different applications, the oldest of which has been running continuously since 2001 and has spawned four variants. Here we describe the Eurasia system and how it meets the challenge of using Web Services for large-scale information management.

Large-Scale Information Management

To increase the probability of high-quality information analysis, users must collect a sufficient number and variety of documents on a given topic. This requires access to diverse Web sources. In addition, users must structure, extract, convert, and visualize the information to meet their specific needs.

Management Challenges

To analyze a particular news topic, for example, several tasks are involved: retrieve news articles from multiple online sources, filter out noise links, classify and rank articles based on predefined topics and place names, compute the cross-product between the two categories, and produce Web portal pages that present the analysis results. In all, users must invoke 21 services to complete this basic analysis task. More advanced applications require more services. As the “Eurasia Applications” sidebar describes and Figure 1 shows, for example, our publicly available GeoTopics news analysis application uses 92 component services (www.isi.edu/geoworlds/geotopics). In all cases, users must ensure that each service they invoke can handle other services’ document structure and semantics. When they detect a mismatch, users must convert or preprocess the data so the service can accept it.

Eurasia Applications

Using Eurasia, we built — in one week — a large-scale news-analysis application, GeoTopics, composed of 92 component services. Among GeoTopics' 92 services, 29 are input and converter services that the service broker semiautomatically selected and inserted into the application. In addition, Eurasia's application composition mechanism hid services' interface details and semiautomatically bound services based on I/O requirement descriptions. Without these semiautomatic selection and binding features, building such a large-scale, complex application would have taken far longer than seven days. Also, without Eurasia's service-coordination mechanism, it would be hard for users to manage the concurrency and synchronization among the 92 services and execute the entire application within a short time.

In addition to the GeoTopics application, we have used Eurasia at a US military site, working with intelligence analysts to develop, adapt, and reuse various Web-based information-management applications, including natural disaster, drug-trafficking, and regional terrorism analyses.

Even for highly trained users who understand a system's functionality and input data requirements, large-scale information management is complex in itself. To improve task quality and performance, however, users must also take advantage of previous success with information-management tasks. To reuse information analysis steps, users must adjust them to the system environment. Some previously available services might become unavailable, and some system resources (processors and memory) might be insufficient to run certain service types. Moreover, handling format changes on information sources can require additional services, such as filters and converters.

Users must also adjust existing information-management steps based on their changing requirements. In some cases, although the overall sequence and types of key information-management steps are the same, users might require a different set of services to meet changing needs. When users must repeat a set of analyses frequently to capture time-sensitive information, for example, they often must replace slow services that produce sophisticated results with faster services whose results are less refined.

Addressing the Challenges

To meet these challenges and help users perform large-scale information-management tasks, we need mechanisms that help users find appropriate services and assemble them coherently for their tasks. These mechanisms should also make it easy to

- resolve mismatches between services,
- repeatedly run the assembled services, and
- reconfigure services to cope with dynamic aspects of Web resources, information-management services, and user requirements.

The proposed Web Services Description Language (WSDL) uses XML to describe Web services as collections of communication end points (ports) capable of exchanging messages.[2] Business Process Execution Language for Web Services (BPEL4WS)[3] and the Defense Advanced Research Projects Agency's Agent Markup Language Service (DAML-S)[4] use WSDL as a framework for providing an XML-based language for coordinating individual Web services to compose high-level business processes.

Developers can use BPEL4WS or DAML-S as an underlying language to describe information-management applications that utilize Web services. However, without high-level tool support, adapting these languages to large-scale information-management tasks is difficult for four reasons.

- It's time-consuming to select services, because the languages offer no support for quickly exploring possible information-management options at each step. Without a mechanism to explore such options based on the target information type, it's difficult for users to recognize which services to incorporate into an application.
- The languages lack an abstraction mechanism, which hinders application reuse and adaptability. Many information-management tasks require similar analysis steps. To reuse these patterns for multiple tasks and adapt the application for different system environments, users must be able to express an information-management application at an abstract level and instantiate and test it by assigning different combinations of service instances. Although BPEL4WS and DAML-S provide facilities to represent abstract processes, we need a mechanism that easily converts between abstract and concrete application representations.
- Service and application transparency is hindered by a lack of collection-based representations. BPEL4WS and DAML-S focus on repre-

senting programming details (data bindings and control structures) rather than high-level processing steps for analyzing and presenting document collections. This makes it difficult for users to understand information-management applications.
- Application reuse and execution are unreliable because the languages lack mechanisms to handle Web resources' dynamic aspects. To reliably reuse and run an application, users should be able to select and coordinate an application's service instances dynamically, based on the availability of information sources and services. Although BPEL4WS and DAML-S offer fault handling and conditional branching mechanisms, they should be incorporated with a high-level tool that lets users monitor services' status and adjust them based on available alternative services and other resources during runtime.

Conventional Web-based information-management systems consist of a client subsystem, which lets users uniformly access information-management services, and a service access infrastructure, which lets higher-level system entities locate and access those services. The service access infrastructure typically uses WSDL and DAML-S, coupled with service registries — such as Universal Description, Discovery, and Integration (www.uddi.org) — to find Web services based on particular constraints and connect to them using the service groundings that the description languages provide.

To solve the four problems described above, we introduced Eurasia as a layer between the client subsystem and the service access infrastructure.

The Eurasia Architecture

Figure 2 shows the Eurasia architecture, which consists of three main components:

- The *component-description framework* provides a taxonomy of services' functionality and I/O requirements (stored in a service metadata repository) that lets users explore different service options and interoperations.
- The *template composer* uses the stored metadata to measure services' interoperability and compatibility, and helps users combine interoperable services to build or refine their applications.
- The dataflow-based *service coordinator* runs the current version of a user's application by controlling the synchronization and parallelism among services based on their data dependency. It also lets users reconfigure the application during runtime.

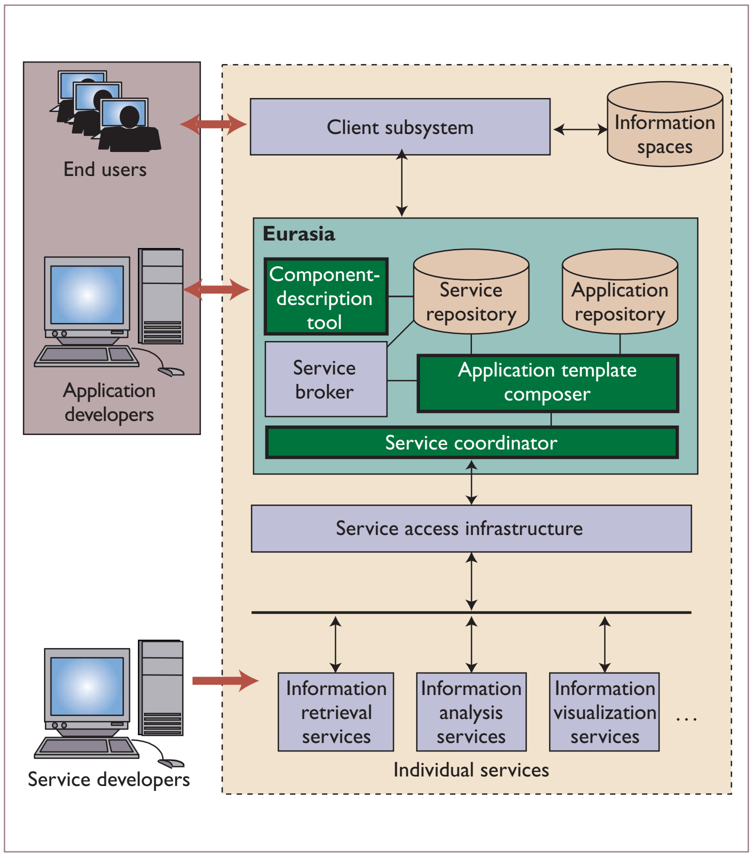


Figure 2. The Eurasia architecture. Using Eurasia, users can explore service options, describe the target document collection, and combine compatible services in a test application.

As Figure 2 shows, users can select and run an application via the client subsystem's user interfaces. The service coordinator then interacts with the service access infrastructure to invoke and control the individual service instances distributed in the network.

We divide Eurasia users into two groups: end users and application developers. The end users are information workers (such as intelligence analysts) who select one of the predefined information-management applications from the application repository for their task, perform some minor adaptation steps (such as substituting services and adding more viewers for results), enter input data, and run the application. They also monitor application execution and, when needed, reconfigure the application during runtime by replacing faulty services. The application developers are more sophisticated users who use the template composer to build task-

Related Work in Large-Scale Software Composition

Software product-line engineering uses a domain-specific architecture and framework to facilitate the rapid production and strategic reuse of software components. In this approach, developers identify and reuse common assets — such as software components, architecture, requirement statements, and specifications — to rapidly create a reliable software products family.[1] Eurasia's application-development paradigm applies this product-line engineering approach to the Web-based information-management domain.

Eurasia's component-description framework and template composer provide the software product line's core assets, producing an application family based on target missions and user groups. The component-description framework lets domain engineers characterize and identify common services for a product family using taxonomies of functionality and I/O document-collection content and structure. Eurasia's service broker lets application engineers explore various options and optimize service selection to build an information-management application for a product family, while the templates define how to assemble those component services.

To leverage the benefits of component-based programming and improve service reusability, Eurasia uses the megaprogramming's programming-in-the-large paradigm for large-scale, heterogeneous, and distributed software components.[2] Megaprogramming uses domain-specific ontologies to abstract and describe the data structures and operations of large software components (megamodules). It builds applications using megamodule interconnections, which it represents using a megaprogramming language. Megaprogramming also uses a megamodule repository and dictionary, which lets developers reuse megamodules for multiple applications.

We added a few features to the megaprogramming paradigm for use in Web-based information-management systems. First, we added a flexible and scalable way of resolving mismatches between services by explicitly inserting intermediate services (converters) instead of making megamodules (services) conform to general schemas defined in an application. Second, we used domain-specific models that describe components and applications; these models are key to Eurasia's collection-based data representation, dataflow-based application composition, and service coordination mechanisms. Finally, we used a template-based application-development mechanism that lets users dynamically instantiate applications based on service availability and reparameterize applications for different tasks. This makes applications reusable for multiple tasks and different system environments.

References

1. D.M. Weiss and C.T.R. Lai, *Software Product-Line Engineering: A Family-Based Software Development Process*, Addison-Wesley, 1999.
2. G. Wiederhold, P. Wegner, and S. Ceri, "Towards Megaprogramming," *Comm. ACM*, vol. 35, no. 11, 1992, pp. 89–99.

oriented information-management applications. They also use the component-description tool to register component services by classifying their functionality and I/O document collections.

As the "Related Work in Large-Scale Software Composition" sidebar describes, the Eurasia architecture provides the core assets of a software product line, which lets users build application families and reuse them for multiple tasks. The architecture also facilitates megaprogramming's programming-in-the-large paradigm, which further leverages component-based programming benefits.

Component-Description Framework

Using the component-description framework, users can find appropriate services by browsing a functional taxonomy and evaluating interoperability with other application services. The framework also semiautomatically identifies and suggests any missing components required to resolve I/O requirement mismatches between services. The framework consists of two basic components: the metadata description model and the service broker.

The metadata description model characterizes and represents the types of document collections and services. One of our goals in designing the metadata model was that it be capable of representing data types at the collection level, rather than at an individual document level. Because document collections are the information processing units in information-management tasks, users tend to compare and select services based on existing document collection types and target collection requirements. Eurasia collects service metadata descriptions in a persistent service repository; these descriptions are currently in our own XML format, but we'll likely convert them to WSDL or DAML-S.

The metadata model enables a simple and lightweight semantic representation. It characterizes services' functionality and I/O data semantics using domain-specific taxonomies; specifically, it represents a document collection's semantics in terms of content and structure types. Let's take, for example, a document collection that characterizes news articles based on references to place names. We would describe the collection's content as a "place-name-based document classification" and its structure as an "acyclic category structure." This dual-taxonomy representation makes the taxonomy hierarchies simpler, makes the reasoning mechanism more efficient, and lets users rapidly narrow the candidate set of services for their tasks.[5] In our experience, the taxonomy specifics do not greatly matter; as long as the system applies them with reasonable consistency, taxonomies help users sort through alternatives, just as a balanced binary tree speeds search without requiring the tree's content.

The framework's context-sensitive service broker matches interoperable and compatible services against a document collection or another service by comparing their functional and I/O data semantics. Given a set of document collections, the service broker compares the collection semantics with available services' input data semantics to identify appropriate services to process the document collections. The broker can also compare a service's output data semantics with another service's input data semantics to identify their semantic interoperability.

Because Eurasia represents metadata at the collection level, we designed the reasoning mechanism to be collection-based and defined a metric to determine the semantic relationship between document collections. The metric measures subsumption relations and semantic distance between I/O document collection sets. This semantic reasoning mechanism lets the service broker return all services that are semantically interoperable or compatible with a document collection or target service.[6] Our news document collection, for example, classifies news based on geographical entities; Eurasia recognizes that the collection could be visualized using both a category viewer and a geographical mapping service. However, Eurasia would rank the geographical mapping service higher because its input semantics is closer to the document collection semantics.

Template Composer

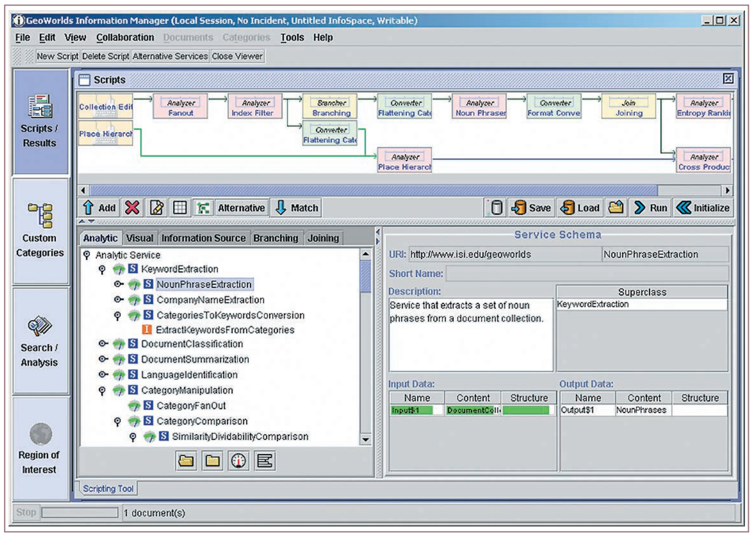
The template composer lets users build information-management applications by describing document collection requirements and transformations. The composer represents transformations of document collections in a dataflow graph, and semiautomatically binds services together based on their I/O relationships. The composer automatically establishes a precedence relationship between two services if one service's output document collection is bound to another service's input document collection. This dataflow-style composition provides a high-level glue-code for combining distributed component services for information-management tasks.

Using the service broker, the composer suggests appropriate transformations for a document collection set and semiautomatically inserts the next set of services in an application. It also automatically identifies and inserts missing components (such as converters and input services) that are required to resolve services' semantic and syntactic mismatches. Once a template is composed, users can easily modify it to perform information-management steps for other similar tasks.

The template composer also lets users exchange their information-management applications with other users. They can exchange a template by sending an XML description and instantiate it by selecting and assigning available service instances in the local system environment. (Our XML-based template description language predates BPEL4WS and DAML-S, but the mechanism it uses to represent data bindings and service coordination structures is similar to that in DAML-S. We are currently investigating how to adapt BPEL4WS or DAML-S to represent application templates.) Once a template is instantiated, the composer creates a local proxy for each component service. This proxy stores interface-level information to access its service instance and hides this information from the high-level application. These system-dependent proxies let users reuse templates in different system environments.[6]

Eurasia's features lower the time and skill required for users to perform complex information-management tasks.

To help users compose an application dataflow graph, we developed a visual template-composition tool (see Figure 3, next page). The tool shows a taxonomy of interoperable services (on the screen's lower left side) and lets users easily select and add them to a dataflow graph at the top of the screen. The screen's lower right section shows detailed descriptions of a service's functionality and I/O data semantics. When the user adds a new service to an application, the tool automatically establishes detailed data bindings between services internally, and shows only the high-level information flows among services.



Service Schema

URI: http://www.isi.edu/geoworlds NounPhraseExtraction

Short Name:

Description: Service that extracts a set of noun phrases from a document collection.

Superclass
KeywordExtraction

Input Data:

Name	Content	Structure
Input$1	DocumentColl	

Output Data:

Name	Content	Structure
Output$1	NounPhrases	

Figure 3. Template composer. As this example from the GeoTopics application shows, users can select services from the taxonomy (lower left) based on detailed descriptions (lower right) and add them to the graph (top of screen).

Figure 3 shows a part the GeoTopics application template. The initial document collection (which the user entered using a collection editor) contains news media homepages (HTML documents). Fanout, a document extraction service, collects articles from each homepage. Fanout's output passes through an index filter that removes static documents, such as weather and stock information pages. Next, the Noun Phraser service runs on each subcollection of news articles within the filtered collection, reorganizing inputs according to a noun-phrase classification. Other services classify documents in parallel on the basis of place names, as the figure shows.

Service Coordinator

Our service coordinator invokes and coordinates distributed service instances, automatically synchronizing them and controlling parallelism. The coordinator also supports a parallelism-control mechanism that can spawn and dynamically merge concurrent service branches based on a document collection's structure.[7] In addition, it can switch service instances during runtime to overcome service faults and other dynamic Web situations.

We incorporated this mechanism with the template composer; this lets users specify coordination actions such as synchronization and concurrent executions among services in an application template. Using the dataflow-style coordination model, the system uses data dependencies to automatically determine an application's synchronization and concurrent execution points. Users can thus focus on describing the transformations among document collections rather than describing detailed control structures among services. In addition, because the dataflow system identifies all possible parallelism and automatically coordinates the distributed services, we can maximize the runtime performance of information-management tasks.

In addition to the implicit parallelism determined by the dataflow system, application developers can explicitly specify parallel service branches. Eurasia's service coordination model supports a service-based control mechanism that lets application developers use explicit control services to specify context-sensitive branching and joining controls. The model itself does not differentiate regular services from control services. The branching and joining services fork and join an input document collection's multiple service branches based on the collection's structure.

This service-based control mechanism lets users describe abstract parallelism among services by considering only the high-level structure of the document collections that services exchange, and makes the application representation simple and easy for users to understand. In Figure 3, for example, the branching service recognizes an input document collection's structure and dynamically creates an independent noun-phrase extraction service branch for each article set from the same news source. The multiple branches that this service spawns run concurrently and are joined by the joining service, which merges results from each noun-phrase extraction.

During runtime, users can reconfigure an application's data flow, which is necessary when they need to switch to a different service or service instance. Replacing a service does not affect the rest of the data flow; the replaced service recovers previous job status and continues its process. This runtime reconfiguration feature makes information-management tasks more reliable.

Discussion and Future Work

Eurasia's features lower the time and skill required for users to perform complex information-management tasks. The exploration and testing features increase application developers' productivity in developing information-management steps, while the capturing and reuse features extend end users' application adaptation and execution capabilities. Eurasia also increases the reliability and efficiency of user-developed Web-based information-management tasks. Because our approach focuses on document collections, it's easier for users to express and understand complex information-management steps, which increases application abstraction level. This task abstraction lets users dynamically instantiate recorded steps in a Web-based system environment. Finally, the dataflow-based service coordinator lets users maximize and dynamically control service parallelism, and reconfigure steps based on resource availability.

Applying Eurasia to practical information-management tasks has helped us identify the need for several additional capabilities:

- *Service exploration based on usage history.* As the number of available services increases, users have an increasingly difficult time determining which services might apply to a document collection type for a specific task. Seeing how other users have applied different services would simplify and accelerate this service-exploration process and make applications more reliable and robust.
- *Immediate feedback in assembling services.* As users develop applications that are complex and handle numerous Web documents, they need immediate feedback — such as sample analysis results — when they add a new service to an application. Such a feedback mechanism would speed the testing of a complex application's services.
- *Collaborative application development and sharing.* As users develop analysis steps for large-scale information-management tasks, being able to collaboratively develop and share applications will reduce redundant efforts.

We are also investigating how to extend Eurasia's models and mechanisms to compose and coordinate applications in other domains, including e-commerce, real-time data management, and hybrid simulations.

Acknowledgments

We thank Barry Boehm, Gio Wiederhold, Pedro Szekely, Nenad Medvidovic, and Ramnath Chellappa for their guidance and comments on this article. We also thank Peter Will, Robert MacGregor, Ke-Thia Yao, and Geoffrey Pike for their careful review. Much of this work was sponsored by DARPA and the US Air Force Research Laboratory, Air Force Materiel Command, under agreements F30602-00-2-0610 and F30602-00-2-0576. The views and conclusions contained herein are those of the authors and do not necessarily represent the official policies or endorsements, either expressed or implied, of DARPA, the Air Force Research Laboratory, or the US Government.

References

1. R. Neches et al., "GeoWorlds: Integrating GIS and Digital Libraries for Situation Understanding and Management," *The New Rev. Hypermedia and Multimedia (NRHM)*, vol. 7, 2001, pp.127–152.
2. E. Christensen et al., "Web Services Description Language (WSDL) 1.1," World Wide Web Consortium note, Mar. 2001, www.w3.org/TR/2001/NOTE-wsdl-20010315.
3. T. Andrews et al., "Business Process Execution Language for Web Services version 1.1," BEA Systems, IBM, Microsoft, SAP AG, and Siebel Systems, May 2003, www-106.ibm.com/developerworks/library/ws-bpel/.
4. A. Ankolekar et al., "DAML-S: Semantic Markup for Web Services," *Proc. Int'l Semantic Web Working Symp.* (SWWS), Semantic Web Organization, 2001, pp. 411–430; www.semanticweb.org/SWWS/program/index.html.
5. K.-T. Yao et al., "Asynchronous Information Space Analysis Architecture Using Content and Structure Based Service Brokering," *Proc. 5th ACM Conf. Digital Libraries* (DL 2000), ACM Press, 2000, pp. 133–142.
6. I.-Y. Ko, R. Neches, and K.-T. Yao, "Semantically-Based Active Document Collection Templates for Web Information Management Systems," *Proc. Int'l Workshop on the Semantic Web*, 2000, pp. 63–72; www.ics.forth.gr/isl/SemWeb/program.html.
7. I.-Y. Ko, K.-T. Yao, and R. Neches, "Dynamic Coordination of Information Management Services for Processing Dynamic Web Content," *Proc. 11th Int'l World Wide Web Conf.*, ACM Press, May 2002, pp. 355–365.

In-Young Ko is a postdoctoral research associate in the Distributed Scalable Systems Division of the University of Southern California's Information Sciences Institute. His research interests are in software component management and coordination mechanisms for large-scale, distributed system environments, focusing on Web-based information management. He received a BS and an MS in computer science from Sogang University, Seoul, Korea, and a PhD in computer science from the University of Southern California. Contact him at iko@isi.edu.

Robert Neches is director of the Distributed Scalable Systems Division of USC's Information Sciences Institute and research associate professor in the USC Computer Science Department. His research interests are in distributed software systems engineering, information management, intelligent human–computer interaction, and computer-supported cooperative work, planning, scheduling, decision making, and decision support. He received his PhD in psychology from Carnegie Mellon University. Contact him at rneches@isi.edu.