University of South Carolina
Scholar Commons

Faculty Publications | Computer Science and Engineering, Department of

2008

Web-Scale Workflow: Integrating Distributed Services

M. Brian Blake

Michael N. Huhns
University of South Carolina - Columbia, huhns@sc.edu

Follow this and additional works at: https://scholarcommons.sc.edu/csce_facpub

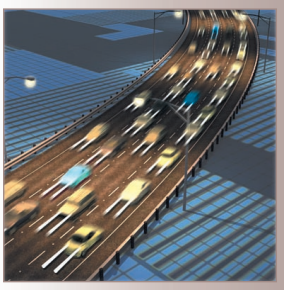
 Part of the Computer Engineering Commons

Publication Info

Published in *IEEE Internet Computing*, Volume 12, Issue 1, 2008, pages 55-59.
http://ieeexplore.ieee.org/servlet/opac?punumber=4236
© 2008 by the Institute of Electrical and Electronics Engineers (IEEE)

This Article is brought to you by the Computer Science and Engineering, Department of at Scholar Commons. It has been accepted for inclusion in Faculty Publications by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.

Editors: M. Brian Blake • *mb7@georgetown.edu*
Michael N. Huhns • *huhns@sc.edu*



Web-Scale Workflow

Integrating Distributed Services

Modular applications, components, and services are all ways of describing the product of an organization's efforts to embody its capabilities in autonomous software modules. In fact, the integration of services using well-established workflow paradigms could amplify an organization's capabilities with the creation of a full-blown, inter-organizational system of systems. This is the essence of Web-scale workflows. Considering the recent popularity and acceptance of service-oriented technologies, the application of such distributed systems is only limited by imagination, but it's also important to understand existing research challenges and their implications to various Web-scale workflow domains.

M. Brian Blake
Georgetown University

Michael N. Huhns
University of South Carolina

Ever since the term *software engineering* was first coined in the 1940s, one of the field's primary goals for software has been modularity, in the form of packaged software mechanisms that perform concise, domain-specific tasks. A significant step forward was the introduction of object-oriented analysis and development, with object-oriented programming debuting in the early 1960s as part of the development of SIMULA 67, the first object-oriented programming language.[1] Interestingly enough, the impact of object-oriented systems wasn't fully realized until the mid 1980s, with the establishment and acceptance of methodologies introduced by Ivar Jacobsen, Grady Booch, and James Rumbaugh. Similarly, although component-based programming appeared in the late 1960s as part of a NATO conference on the software crisis,[2] it wasn't until the mid 1990s that large-scale applications came to the forefront.

More recently, the combination of components and the World Wide Web has led to the emergence of Web services.[3] Systems constructed out of Web services work according to the principles of service-oriented computing,[4] an area that has experienced a relatively short horizon from conceptualization to application. Although Gartner Research is credited with the first mention of a service-oriented architecture in 1996,[5] just five years later, research labs, industrial organizations, and federal government facilities all moved to create these types of architectures in their own domains.

Web-scale workflows are possible due to services that exist both inside and outside an enterprise. Properly developed Web services should execute

1089-7801/08/$25.00 © 2008 IEEE

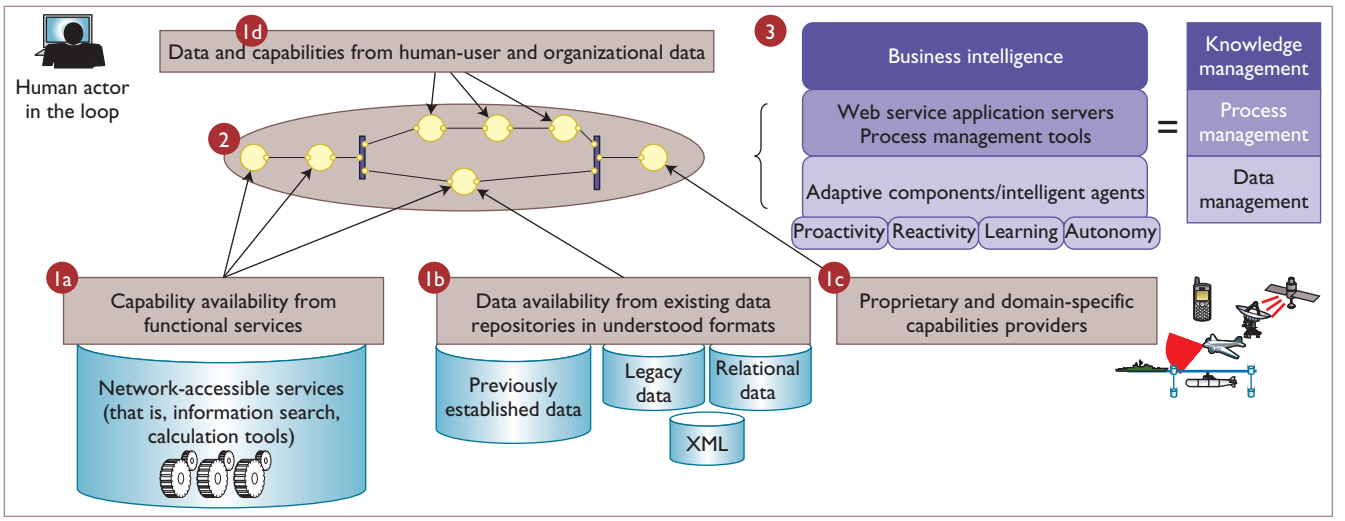


Figure 1. Combining capabilities in a Web-scale workflow. A variety of services made visible in step 1 are combined into the interorganizational process in step 2. The principled management of service-centric systems leads to a modular infrastructure that promotes business intelligence (step 3).

well-defined tasks as supported by concise, openly accessible interfaces. As such, the first step of any Web-scale workflow architecture should be the acceptance of a service-based development approach[6] and the population of an array of services that represent capabilities across a wide variety of domains. This article — indeed, this entire track — examines methodologies, processes, and leading techniques for the realization of such infrastructures in industry, government, and societal domains.

Web Services

Services can help realize many types of business needs (sections 1a through 1d in Figure 1 show some examples), but the most common Web services on the Internet today are capabilities that let clients

- convert data from one format to another (such as Fahrenheit to Celsius),
- perform simple table look-up (such as census data),
- retrieve real-time information (such as stock quotes or weather updates), and
- make simple combinations of retrieval and conversion tasks (such as converting dollars to yen).

Web-scale workflow paradigms dictate that these types of services be replicated in the generation of highly specialized transformation tasks. Perhaps the biggest challenge to government organizations and other large-scale enterprises is the sheer magnitude of legacy software components and data repositories that must be integrated into service-based paradigms. The major task for such areas is how to wrap these types of data and capabilities with service-based interfaces.

Services can also help us expose data and information from open and proprietary devices, such as sensors, mobile devices, transportation systems, and other communication systems. Additionally, many human-enacted processes and systems can be triggered with service-based communication. In many cases, human-enacted systems combine human-based tasks with software and system-oriented operations.

Although defining services is the first step, the major challenge for Web-scale workflow environments is the ability to organize capabilities into higher-level inter-organizational processes (see section 2 in Figure 1). Often, software system architects can define these processes by using well-established concepts such as *workflow* and *supply chain*, but in managing these processes, organizations must maintain peripheral functions to meet nonfunctional concerns (such as performance, reliability, and security). Any mechanism that controls Web-scale workflows must be both reactive and proactive by learning to act autonomously as the environment changes.

Recent trends in distributed computing[7] suggest that adaptive software components or software agents could act as proxies for managing these processes. When agents are allowed to access and interact in Web service infrastructures, system integrators can use the resulting infor-

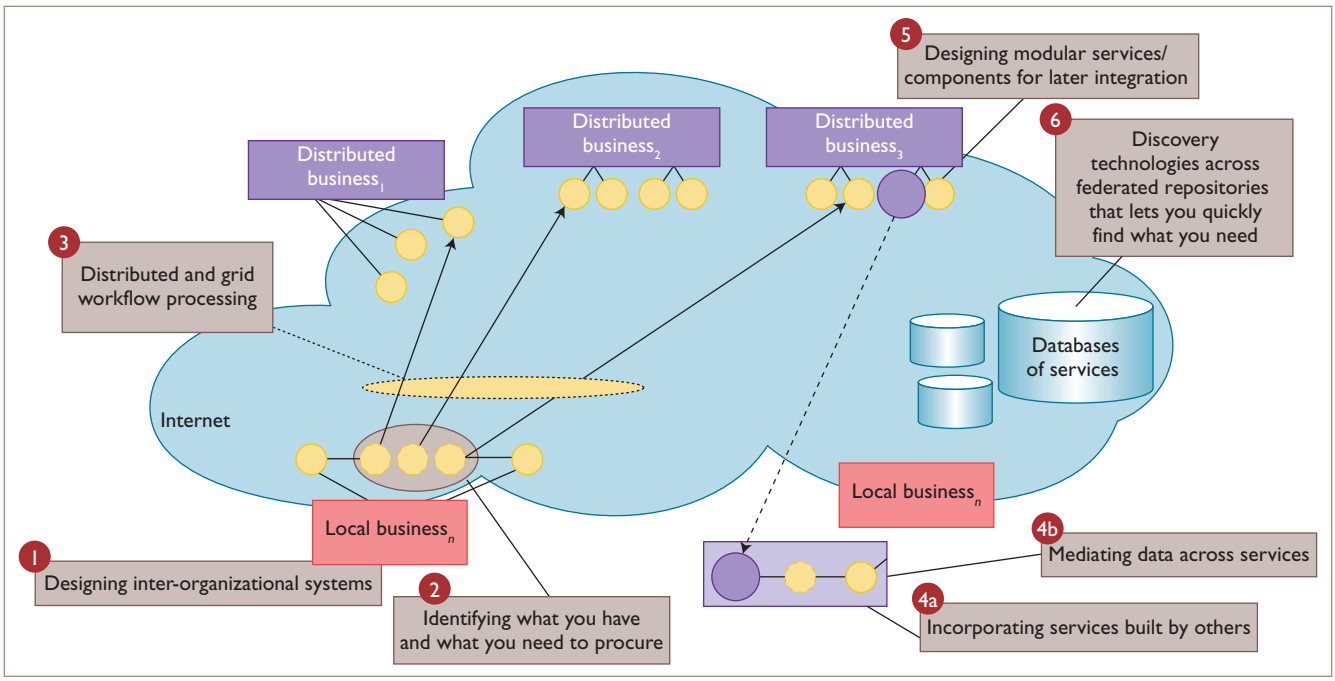


Figure 2. Research issues related to Web-scale workflow environments. These open research areas cover the operation of a Web-scale workflow from software architecture and design to real-time system operations.

mation and metrics to enhance operations across an entire virtual enterprise. The integration of agents to understand business intelligence in a service-oriented environment exemplifies how data management and process management combine to facilitate knowledge management in a virtual enterprise.

Open Research Issues

Although the application of intelligent agents to service-oriented environments has its own issues,[8] open research questions also surround the more general Web-scale workflow environment, as Figure 2 shows. A Web-scale workflow environment is composed of freely available services made network accessible by multiple organizations, even though some organizations have partnerships and others don't. Several enabling activities are required for this environment to operate in an ad hoc, on-demand manner.

Designing Inter-Organizational Systems

How does a system architect design a system composed of services distributed across multiple organizations? Because Web-scale workflow systems are hierarchical by nature, the major issue is visualizing the system to facilitate proper analysis. Leading approaches suggest that visual modeling is perhaps the most appropriate approach when architecting systems, but a gap exists between the leading approaches to model-driven architecture (MDA)[9] and the specific requirements necessary in Web-scale workflow environments. In the latter, the distribution of functional capabilities is a required aspect that isn't well-represented in MDA approaches, which weren't initially developed with services in mind. Consequently, current efforts are under way within the Object Management Group (OMG) to align business process management with MDA. System integrators must interpret the design at runtime in a very dynamic environment — the Web — which can lead to problems with service discovery.

Interpreting and Discovering Capabilities

Although traditional software engineering life cycles suggest that requirements can be elicited via interviews, observations, and human interaction, such approaches are ineffective for services advertised with standard WSDL interface specifications. Semantic specifications of interfaces help facilitate the definition and understanding of services in a particular domain context, but most system developers haven't uniformly adopted these approaches, and the overarching ontologies tend to be unwieldy (www.daml.org/owl-s). Furthermore, inter-organizational consensus of the core semantics tends to be difficult to achieve. Agile, user-friendly techniques and tools are needed

to facilitate the definition and understanding of services, whether local or external.

Distributed and Grid Workflow Processing

In some cases, organizations have pre-established relationships in which services for collaboration are well understood. In an open environment, these services might be distributed across different organizations, with services invoked on machines of various bandwidths and processing power. An opportunity arises when organizations in this distributed environment can choose between similar capabilities that have different quality-of-service expectations. This type of environment requires monitoring and control protocols that can efficiently manage the composition of services while optimizing the overall process's anticipated response time.

Adopting and Consuming Distributed Services

Service *consumption* occurs when an organization is able to integrate another organization's capability into its own processes — here, the service remains on the provider's system. Service *adoption* is somewhat different — here, providers develop services that consumers can download and host themselves. Adopted services typically connect to capabilities that the original provider offers; Amazon has popularized this paradigm by allowing third-party vendors to sell goods directly. When services are either consumed or adopted, the consumer must be able to manage inputs and outputs to the consumed services from local, perhaps pre-existing, services. Because both local and external services are probably coming from different organizations with different intentions, a challenge is how to mediate potential mismatches.

Designing Services for Adoption

Designing and developing services for adoption requires developers to use flexible domain- and system-independent technologies. An effective adoptive service should be capable of redeployment in systems implemented in a range of programming languages or operating systems. A challenge in this area is how to develop sandbox environments that allow for the smooth transition of services from one system to another. The emergence of applications for virtual machines and virtual servers could assist with the notion of sharing adoptive services.

Federated Service Storage and Discovery

The leading standard for storing, advertising, and discovering Web services is universal description, discovery, and integration (UDDI). In practice, however, it isn't widely deployed for sharing services because it isn't intuitive, even to experienced developers, nor does it support sophisticated search requirements. Furthermore, a federation of UDDI registries, particularly across vendors, isn't defined. As such, many open issues remain before UDDI registries can support on-demand search for relevant capabilities.

Managing Workflow Systems

The overarching problem that aggregates all the others listed in this section is how to manage Web-scale workflows: components might be outside an organization's control. Researchers are actively studying techniques for solving this problem, including how to use substitutable services to replace ones that misbehave and how to use proxy services to shield the workflow from any failures or changes in external service components.

Emerging Applications

Success in developing Web-scale workflow systems might lead to the introduction of several new applications.

Workflows for Scientific Computing

The use of distributed processing and, more recently, grid computing for scientific applications is commonplace, but such implementation architectures are typically conceived for a particular use, with underlying components developed for a domain-specific purpose. Web-scale workflow paradigms can promote collaboration among scientific research groups that haven't previously considered sharing capabilities and resources. This paradigm also promotes interoperation over the Web, with stakeholders able to post and discover relevant services. Similarly, grid users might be able to search for available resources that let them execute services more efficiently. An extension of Web service registries might allow browser-based access to available services, whether they're application-specific capabilities or computational resources.

Service Mashups

A service mashup is a new concept in which

system integrators use Web service provisions to develop integrated capabilities in an ad hoc fashion (for example, the output of a map Web service enhanced with real-time weather and automobile traffic information). Although a workflow implies that services combine in a specific order in a process, mashups promote the aggregation of many data services simultaneously. However, service mashups still require process and policy guidance to respect the priority with which information should be combined. In some cases, privacy policies must be preserved and incorporated into a process.

Automated Service Contracting

As stakeholders begin to release their capabilities openly as Web services over the Internet, consumers will need automated approaches for purchasing these capabilities. In addition, the acquisition and use of services must be bound by a mutually agreeable service level and price. At times, these agreements must be negotiated and ratified. Because access to Web services can be automatic and on demand, the negotiation of service-level agreements should also be automated.[10]

Fortuitously, the Web itself provides a means for solving some of the problems it causes. Enterprises might not be able to manage external workflow components, but they might be able to find alternative services on the Web. When competing service components disagree, a workflow can use voting techniques to determine consistent results. If potential service providers are untrustworthy, a system could use a Web-based reputation network to assess credibility. These examples rely on control mechanisms for service-oriented processes across organizations, mission-oriented semantic and syntactic approaches to service composition, adaptive (perhaps agent-oriented) approaches to Web service workflows, software architecture and engineering approaches for service mediation, and data-management approaches for services (such as innovative service repository and metadata management). Subsequent articles in this track will address these and other challenges. Please read more about submissions at www.computer.org/portal/pages/internet/content/cfptrack.xml.

Acknowledgments

We acknowledge the fruitful conversations with Paul Buhler of the College of Charleston that helped enhance the contents of this article.

References

1. K. Nygaard, "Basic Concepts in Object Oriented Programming," *ACM SigPlan Notices*, vol. 21, no. 10, 1986, pp. 128–132.
2. P. Naur and B. Randell, *Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968*, Scientific Affairs Division, NATO, 1969, pp. 138–155.
3. M. Bichler and K.J. Lin, "Service-Oriented Computing," *Computer*, vol. 39, no. 3, 2006, pp. 99–101.
4. J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," *Semantic Web Services and Web Process Composition*, LNCS 3328, Springer-Verlag, 2005, pp. 43–54.
5. *Service Oriented Architectures, Part 2*, SSA Research Note SPA-401-069, Gartner Research, 12 Apr. 1996.
6. M.B. Blake, "Decomposing Composition: Service-Oriented Software Engineers," *IEEE Software*, vol. 24, no. 6, 2007, pp. 68–77.
7. J.M. Vidal, P. Buhler, and C. Stahl, "Multiagent Systems with Workflows," *IEEE Internet Computing*, vol. 8, no. 1, 2004, pp. 76–82.
8. M.H. Huhns et al., "Research Directions for Service-Oriented Multiagent Systems," *IEEE Internet Computing*, vol. 9, no. 6, 2005, pp. 65–70.
9. J. Mukerji and J. Miller, *MDA Guide Version 1.0.1*, tech. report omg/2003-06-01, Object Management Group, 2003; www.omg.org/docs/omg/03-06-01.pdf.
10. L. Zeng et al., "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, 2004, pp. 311–327.

M. Brian Blake is an associate professor and chair in the Department of Computer Science at Georgetown University. His research interests include service-oriented computing and enterprise integration. Blake has a BS in electrical engineering from the Georgia Institute of Technology and a PhD in information and software engineering from George Mason University. He is a senior member of the IEEE. Contact him at blakeb@cs.georgetown.edu.

Michael N. Huhns is the NCR professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology. His research interests are in multiagent systems, service-oriented computing, ontologies, and — of course — workflows. Huhns has a PhD in electrical engineering from the University of Southern California. He is a fellow of the IEEE. Contact him at huhns@sc.edu.