Realizing Service-Centric Software Systems

Service-centric software systems are an important multidisciplinary paradigm concerned with software systems that are constructed as compositions of autonomous services. These systems extend the service-oriented architecture paradigm (see the “Service-Oriented Architecture” sidebar) by focusing on the design, development, and maintenance of software built under SOAs. The autonomous services of these systems are platform-independent computational elements that can be described,
published, discovered, orchestrated, and programmed using standard protocols to build networks of collaborating applications distributed within and across organizations.

Over the past few years, service developers, integrators, and providers have collaborated to address various challenges to realizing service-centric software systems. Their efforts have contributed to service engineering, discovery, composition, integration, monitoring, and management. Results include new processes, methods, and tools to assist software engineers in developing and deploying complex, dependable service-centric systems. However, the service-centric paradigm is evolutionary, so many challenges remain to fulfill its vision.

To achieve adaptable, flexible, interoperable, and maintainable service-centric software systems, we must define processes and create methods, tools, and techniques to support cost-effective application development and use. We must also provide interoperability among the tools and develop adequate approaches to support development life cycles; service discovery and composition; service deployment, binding, and monitoring; quality of services and security; service-level agreements; and service description languages.

The articles

In this issue, we present five articles that tackle service-centric software systems.

In “A Domain-Configurable Development Environment for Service-Oriented Applications,” Philippe Lalanda and Cristina Marin describe two tools to support service composition. The first supports application specification on the basis of syntactic descriptions of concrete services. The second allows domain-specific environments to be developed according to rules specified by domain experts. The authors report lessons learned using these tools to develop home Internet gateways in the power-distribution industry.

Supporting adaptation is an important characteristic of service-centric software systems. In “PAWS: A Framework for Executing Adaptive Web-Service Processes,” Danilo Ardagna, Marco Comuzzi, Enrico Mussi, Barbara Pernici, and Pierluigi Plebani describe a flexible approach for executing Web services. They base their approach on business processes that support adaptation both at design time and runtime, and they include several case study applications.

An important aspect of service-centric software systems is the identification of correct services. In “Improving Web Service Discovery with Usage Data,” Aliaksandr Birukou, Enrico Blanzieri, Vincenzo D’Andrea, Paolo Giorgini, and Natalia Kokash present a system for improving service discovery on the basis of data provided by service clients rather than service owners. The approach identifies services according to prior decisions made in similar situations. The article presents evaluation results according to two similarity metrics.

The Semantic Web plays an important role in service-centric software systems. In “SESA: Emerging Technology for Service-Centric Environments,” Tomas Vitvar, Michal Zaremba, Matthew Moran, Maciej Zaremba, and Dieter Fensel describe an architecture to support service integration and interoperation on the basis of semantic languages and service models. The article also surveys Semantic Web technology for Web services.

In “Decomposing Composition: Service-Oriented Software Engineers,” M. Brian Blake proposes an incremental development approach that integrates software engineers with subject matter experts as a first step toward development environments for service-centric software systems. Different stakeholders must have different roles in the development process, and Blake identifies two in his article: service-oriented software engineer and service-oriented software integration engineer. The first role is concerned with defining business processes that...
are then realized by a service composition; the second, with defining service-level agreements and runtime compositions based on these SLAs.

In the Point-Counterpoint discussion, Donald Ferguson and Richard Veryard debate the roles of tools versus standardized architectures and design guidelines to support the industrial transition from traditional to service-centric software systems.

We hope you find the ideas and experiences described in this issue helpful in developing service-centric software systems. To learn more, consider participating in various groups listed in the “Service-Centric Software Systems Groups” sidebar. We also hope this special issue will stimulate further technology development and transfer in the service-centric software system community.

Acknowledgments

We thank the reviewers for devoting their time and expertise to help us review all the submissions and select the articles in this issue.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.

About the Authors

Olivier Nano is a lead researcher and software design engineer at the European Microsoft Innovation Center in Aachen, Germany. His research centers on software technology transfer from concept to production. His current focus is on service-level agreements in service-centric systems—specifically, on languages, tools, and protocols for autonomic service-level agreement. He received his PhD in computer science from the University of Nice—Sophia Antipolis. Contact him at EMIC, Ritterstrasse 23, 52074 Aachen, Germany; onano@microsoft.com.

Andrea Zisman is a reader in the Department of Computing of City University, London. Her research centers on automated software engineering and service-oriented computing. Her current focus is service discovery and composition during development and execution time of service-centric software systems. She received her PhD in computer science from the Imperial College of Science Technology and Medicine. Contact her at City University, Northampton Sq., London E1V 0HB, UK; a.zisman@soi.city.ac.uk.