THE SERVICE-ORIENTED COMPUTING PARADIGM

Service-Oriented Computing (SOC) is a new computing paradigm that utilizes services as the primitive computing unit to support development and composition of larger-granularity services in heterogeneous environments. The specification and execution of services in such environments is meant to support agility in application structures, flexibility for business processes and development of applications that can easily span organizational boundaries. The vision is being realized with the standardization (Umapathy & Purao, 2007) of several core layers for Web Services (WS) as well as WS-* components that provide important add-ons and respond to non-functional requirements such as reliability (Erl, 2005). The overall infrastructure is being built with these components and is called Service-Oriented Architecture (SOA). While SOA, e.g., Sun’s Jini (Arnold, O’Sullivan, Scheier, Waldo, & Wollrath, 1999) and SOAP (Simple Object Access Protocol)/UDDI (Universal Discovery Description and Integration) (Box et al., 2000), is a relatively new term, technologies related to SOA, e.g., RPC (remote procedure call) (Birrell & Nelson, 1984), have been studied extensively since early 70s. OASIS (Organization for the Advancement of Structured Information Standards) defines SOA as a collection of best practices, principles and patterns related to service-aware, enterprise-level, distributed computing. Together, SOC, WS and SOA have tremendous potential to reshape businesses, including the health and government sector (Zhao & Cheng, 2005) as well as sciences (Foster, 2005). In principle, the service-oriented approach, independent of specific programming languages or operating systems and implemented using a self-describing interface, allows organizations to expose their core competencies programmatically over the Internet or a variety of networks using standard XML-based languages and protocols (Papazoglou, Traverso, Dustdar, & Leymann, 2008). The evolving platform has spawned, over the last few years, tracks, mini-tracks as well specialized conferences such as IEEE Services Congress, International Conference on Web Services and Service-oriented Computing Conference, and has provided an impetus to considerable activity in several research communities, including computer science, software engineering, workflow and process modeling and organizational IT – broadly construed as computation-focused, software-focused and organization-focused.
RESEARCH COMMUNITIES
 AND CHALLENGES

The computation-focused research communities are largely concerned with problems related to composition and execution. Problems of interest to this community, therefore, often include service composition in the presence of multiple services, reliability and efficient execution. The shift to SOA and WS – that includes loosely coupled services – from the product-oriented software architecture can invalidate many techniques developed for traditional software, including the software reliability models. For example, a Service-Oriented software Reliability Model (SORM) evaluates the reliability of WS in two steps, both of which are dynamic and at run-time (Tsai et al. 2004). Another example of an effort in this stream is supporting a wide variety of communication patterns over multiple transport protocols. Papazoglou and van den Heuvel (2007) build a distributable communications and integration backbone for SOA as an integration platform that employs Web services standards and supports a wide variety of communications patterns over multiple transport protocols.

In contrast, the software-focused research community is interested in conceptualizing, constructing and evaluating methodologies and toolsets that individual designers and teams can use for developing and deploying information systems solutions following the service-oriented paradigm. This community has focused on designing service interfaces from existing components by addressing specific analysis and design principles. For example, Papazoglou and Yang (2002) present a design methodology for self-contained, modular, extensible and reusable web services that provides service design guidelines based on the principles of minimizing coupling and maximizing cohesion and van der Raadt, Gordijn, and Yu (2005) describes a framework for the analysis of automated business interactions among different organizations via web services.

The organization-focused research community seeks inspiration from problems in organizational and inter-organizational processes. For example, it deals with workflows and business process concerns that can, at times, overlap with the previous two communities but takes as its point of departure the central role of processes for organizational and inter-organizational operations. Concerns such as process decomposition, service identification, non-functional requirements and service interactions dominate the agendas of researchers in this community with a view to addressing problems observed in industry. Examples of work include composition of services to form new, aggregate services that is part languages such as BPEL4WS (Business Process Execution Language for Web Services) and BPML (Business Process Modeling Language), modeling reusable interaction patterns and providing key enablers for application of service-oriented computing including third-party aggregation of existing services into compositions that, in turn, can be used as services themselves. Other researchers in this community also deal with concerns such as internetworked and inter-organizational Information Systems (IS) that must respond to service level agreements, service infrastructures, and design of elements such as the master data layer and governance concerns (Chen, Zhou, & Zhang, 2006; Erickson & Siau, 2008; Zhao, Tanniru, & Zhang, 2007).

CONTENTS OF THIS ISSUE

In spite of the pervasiveness of concerns and significant research activity, relatively few scholarly papers have been devoted to an investigation of this area within the IS domain. The purpose of this special issue was to provide a forum where such work may be brought together and made available to the IS audiences allowing multiple research communities described above to participate in the discourse. The call for the special issue attracted papers that from the first two research communities with significant overlaps with the
third community. The review process for the special issue consisted of two rigorous rounds of review (aided by a capable editorial board) including a check provided by an industry advisory board that consisted of the following individuals: Michael Kuhbock from the Integration Consortium; Robert Rosko from Unisys North America; Alex Jeyschune, formerly from EMC; and John Semmer from Accenture. The final set of four papers contained in the special issue represents a cross-section of concerns from the above communities.

The first paper, by K. Vidyasankar and Gottfried Vossen, deals with multi-level modeling of web service compositions with transactional properties. It presents and studies a multi-level service compositional model that perceives service specification as going through several levels of abstraction. It starts from transactional operations at the lowest level, and abstracts these into activities at higher levels that are close to the service provider or even the end user. The authors believe that service composition should be treated from a specification and execution point of view at the same time, where the former is about the composition logic and the latter about transactional guarantees. Consequently, their model allows for the specification of the number of transactional properties such as atomicity and guaranteed termination at all levels. They discuss several ways of achieving the composition properties as well as implications of the model.

The second paper, by Dickson K. W. Chiu, Qing Li, Patrick C. K. Hung, Zhe Shan, S. C. Cheung, Yu Yang, and Matthias Farwick, deals with service composition and interaction following separation of concerns with flows and views. They propose a methodology to decompose complex process requirements into different types of elementary flows, such as control, data, exception, and security arguing that a subset of each type of flows (i.e., flow views), necessary for the interactions with each type of collaboration partners, can be determined in each service. They suggest that these subsets collectively constitute a process view, based on which interactions can be systematically designed and managed for system integration through service composition. They illustrate how this middleware, called FlowEngine, implements and manages the flows with web services technologies and describe a case study in an e-gov environment to demonstrate how their methodology can facilitate organizations to design their own services-based integration infrastructures, and how new cross-organizational information systems can be formulated over such an infrastructure.

The third paper by Sami Bhiri, Walid Gaaloul, Claude Godart, Olivier Perrin, Maciej Zaremba, and Wassim Derguech aims at ensuring customized reliability of composite web services. They point out that unlike process components, web services are defined independently from an execution context. The inherent autonomy and heterogeneity that such definitions engender make it difficult to reason about the behavior of composite services, especially in case of failures. They define reliable composition as the outcome where all instance executions are correct from a transactional and business point of view. The approach they devise to ensure reliable web service compositions, therefore, combines the expressiveness of workflow models and the reliability of advanced transactional models. They claim that the approach offers flexibility to the designers in terms of specifying control structure and execution correctness because the specifications include recovery mechanisms to ensure correct execution according to their requirements.

The fourth paper, by Hiroshi Wada, Junichi Suzuki, and Katsuya Oba, is aimed at leveraging early aspects in end-to-end model-driven development for non-functional properties in SOA. The authors suggest that retaining reusability of services is an important goal for SOA infrastructures, and to do this, it is important to separate non-functional properties of applications (e.g., security and reliability) from their functional properties. They propose and examine an end-to-end model-driven development framework that separates non-functional properties from
functional properties and consistently manages them from high-level business processes to low-level implementation configurations. The framework contains two components: a programming language for a new per-process strategy to specify non-functional properties, and a graphical modeling method to define constraints among non-functional properties. The programming language uses the notion of aspects to specify a set of non-functional properties that crosscut multiple services in a business process. The graphical modeling method leverages the notion of feature modeling to formally define constraints among non-functional properties such as dependency and mutual exclusion constraints. The authors describe design details of both and demonstrate their use in developing service-oriented applications along with an empirical evaluation that shows the techniques are cost-effective and efficient.

We hope that the four papers in the special issue will provide a useful cross-section of activity in these research communities, and, in turn, pointers to new research ideas for the IS scholars.

Sandeep Purao  
Vijay Khatri  
Brian Cameron  
Guest Editors  
JDM

REFERENCES


