Research Challenges on Engineering Service-Oriented Applications

E. Di Nitto*, D. Meiländer, S. Gorlatch†, A. Metzger‡, H. Psaier, S. Dustdar§, M. Razavian, D.A. Tamburri, P. Lago¶
*Politecnico di Milano, Italy
dinitto@elet.polimi.it
†University of Münster
Germany
{d.meil, gorlatch}@uni-muenster.de
‡University of Duisburg-Essen
Germany
andreas.metzger@paluno.uni-due.de
§Technical University of Wien
Austria
{H.Psaier, dustdar}@infosys.tuwien.ac.at
¶VU University Amsterdam
The Netherlands
{m.razavian, d.a.tamburri, p.lago}@vu.nl

***Abstract*—This paper focuses on providing an overview of the research challenges that have been identified toward the end of the S-Cube network in the area of service engineering. These challenges concern the need for agility and dynamicity of the development process for service-based applications, the importance of focusing on proper approaches to support migration of legacy application into service-based applications and the role of humans and of teams of humans in service-based applications.**

***Keywords*-Engineering of service-based applications, agile approaches, self-adaptation, evolution, Agile Service Networks, human-provided services, real-time applications.**

I. Introduction

When referring to *Service Oriented Architecture* (SOA) as a paradigm, SOA typically constitutes a set of guiding principles for building *Service-Based Applications* (SBAs). Thanks to these principles, services can be (re-)used in many different settings and service-based applications can meet the requirements for dynamism and flexibility.

SOA enables us to build software systems with a high degree of flexibility. Software services separate ownership, maintenance and operation from the use of the software. Service users thus do not need to acquire, deploy and run software, because they can access its functionality from remote through service interfaces. Ownership, maintenance and operation of the software remains with the service provider [1].

Third-party software services enable organizations to flexibly outsource business functions (typically commodity functions) and to focus on the innovative functions, which differentiates one organization from another.

Thus, SOA promises huge benefits in terms of dynamism and flexibility. However, service-based applications also need to become resilient to their services changing, disappearing or violating their expected quality. Especially in the case of third-party services, the service users do not have control over such changes, thus, calling for novel solutions to address this new dimension of changes.

One of the main missions of S-Cube, the European Network of Excellence on software, services and systems[1] has been to investigate and support the flexibility and dynamism of SOA. In particular, the network has defined a life-cycle that highlights the central role of *evolution* and *adaptation* for service-based applications and has experimented the usage of the life-cycle not only in the development of typical, business-oriented applications, but also in the context of real-time applications deployed on the Cloud. This roadmap summarizes the life-cycle in Section II and highlights some challenges that we have identified while applying it to the real-time application case (Section III). Moreover, the paper elaborates on the importance of agility in the evolution and adaptation of SBAs (see Section IV). It also focuses on the type of support research should offer to the migration of legacy applications into SBAs (see Section V), and introduces humans-offered services as an integral part of a SBA by highlighting the issues that arise in this case (see Section VI). Finally, the paper brings the idea of humans participating into a SBA to the extreme and suggests that heavily human-based processes, such as Global Software Development, could be seen as executed by *Agile Service Networks* where humans are the main executors of services (see Section VII).

[1]http://www.s-cube-network.eu/

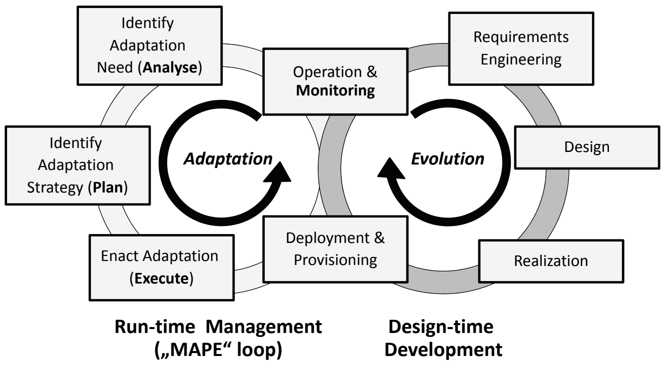
Identify Adaptation Need (**Analyse**)
Operation & **Monitoring**
Requirements Engineering
Identify Adaptation Strategy (**Plan**)
Adaptation
Evolution
Design
Enact Adaptation (**Execute**)
Deployment & Provisioning
Realization
Run-time Management („MAPE" loop)
Design-time Development

Figure 1. The S-Cube Service Life-Cycle Model

II. THE S-CUBE LIFECYCLE

The S-Cube service life-cycle model [2] defines the relevant activities for self-adaptive service-oriented systems and integrates these into a coherent framework. It consists of the following two loops, which are typically executed in an incremental and iterative fashion (see Figure 1):

1) The *Evolution loop* builds on the more traditional development and deployment activities, including requirements engineering, design, realization, and deployment. However, it extends those activities with "design-for-adaptation" steps, such as to define and implement how the system should monitor and modify itself when entering the left-hand side of the life-cycle (e.g., see [3]).
2) The *Adaptation loop* explicitly defines activities for autonomously addressing changes during the operation of service-oriented systems. The activities in the adaptation loop follow the steps of the MAPE loop (Monitor-Analyze-Plan-Execute), which is typically found in autonomic systems [4].

It should be noted that in some cases also adaptation requires human intervention. This is often called human-in-the-loop adaptation [2]. Human-in-the-loop adaptation is different from evolution in the sense that the activities performed by the humans and the artifacts that are modified differ; e.g., the change of requirements documents certainly requires to go through the evolution loop, while the choice between two possible candidate services can be performed as human-in-the-loop adaptation.

III. ADOPTING THE S-CUBE LIFECYCLE TO SUPPORT REAL-TIME PROVISIONING OF ROIA IN THE CLOUD

An important application service area which benefits from the design-for-adaptation approach is concerning Real-Time Online Interactive Applications (ROIA), like online games, interactive e-learning, etc. In ROIA, there are typically multiple users who access a common application state and interact with each other concurrently within one virtual environment. The users connect to the application from different client machines and interact with other users. ROIA are highly distributed applications with challenging QoS demands, such as: short response time to user actions (about 0.1-1.5 s), high update rate of the application (up to 50 Hz), large and frequently changing number of users in a single application instance (up to 10^4 simultaneously). Because of the very high performance requirements of ROIA, the application processing is performed on multiple servers.

In our previous work [5] we have shown that incorporating design-for-adaptation activities into the ROIA development process as proposed by the S-Cube Lifecycle Model helps application developers to address these challenging QoS demands. At the same time, Cloud Computing opens new opportunities for ROIA to serve very high numbers of users and still comply with QoS demands by leasing Cloud resources on demand, particularly for offloading computations from mobile devices to more powerful Cloud resources. Despite a variable number of users, Cloud resources can be used efficiently if the application supports adding/removing resources during run-time. Hence, using Cloud Computing for resource provision and the Lifecycle model for implementing adaptable ROIA complement each other.

In the following, we briefly discuss three main challenges for adaptable ROIA development and provisioning, aiming at Clouds and mobile devices.

A. ROIA Adaptation Along the S-Cube Lifecycle

In order to adapt ROIA to changing workloads, application developers need to implement suitable monitoring and adaptation mechanisms. Our work on ROIA development along the S-Cube Lifecycle Model identified three main scenarios that require adaptation [5]: (i) Change in Quality of Service, e.g., QoS violations caused by unreliable Cloud resources; (ii) Change in computational context, e.g., increased costs for calculating the application state caused by increasingly frequent interactions between users; (iii) Change in business context, e.g., changing user preferences, e.g., many new and concurrent user connections at the same time.

Hence, an important research area is the integration of high-level monitoring and adaptation mechanisms into existing development platforms for ROIA, like the Real-Time Framework (RTF) [2]. In particular, monitoring mechanisms need to provide information about hardware utilization, application processing and future user load in order to address all three types of changes described above.

Furthermore, we identified the demand for balancing run-time adaptation and application redesign. Particularly, the continuous adaptation on Cloud resources may lead to an inefficient application structure which requires redesign.

B. ROIA Provisioning on Cloud Resources

In our previous work [6], we showed that the performance of ROIA can be strongly affected by the virtualization of

[2]http://www.real-time-framework.com

Cloud resources. In order to combine physical and virtual resources (from potentially multiple Cloud systems) for ROIA provisioning, it is important to investigate the impact of virtualization on ROIA and develop prediction models for ROIA performance on different Cloud systems.

Cloud platforms offer services that provide monitoring and adaptation mechanisms for their Cloud resources, e.g., Amazon Cloud Watch or Amazon Elastic Load Balancing [3]. However, these services only provide generic system information about resource utilization (CPU, memory, bandwidth, etc.). This information is not sufficient for up- and down-scaling of ROIA sessions since ROIA have a specific runtime behaviour: e.g., regardless of the current number of users, an online game may run with a constant CPU load of 100% in order to deliver the highest state update rate possible.

Since Cloud resources may require a long startup time (up to several minutes), proactive resource management is particularly important for ROIA provisioning on Clouds in order to address changes in computational and business context. For example, neural networks can be used for predicting the variable user load. Hence, another important research area is the development of proactive resource management strategies for ROIA in Cloud environments that take into account real-time communication QoS and the impact of Cloud virtualization on latency, jitter, throughput, fragmentation, etc. Furthermore, cost-effective resource management requires mechanisms for resource buffering that address static leasing periods of Cloud resources.

C. *ROIA on Mobile Devices*

Mobile devices are highly desirable for many ROIA in order to improve the mobility and interactivity between users in virtual worlds and augmented reality. However, the comparatively low CPU and memory capacity, as well as limited battery life of mobile devices, combined with potentially non-reliable, high-latency wireless networks for communication imposes new challenges.

Since mobile devices have limited CPU and memory capacities, application processing should be offloaded to more powerful Cloud resources. Hence, application developers need transparent high-level development tools that take into account real-time sensitive data streaming, congested mobile access links, roaming communication endpoints and migration, switch or aggregation of streaming sources.

IV. Agility in Evolution and Adaptation of Service-based Applications

When the system evolves, it goes through a reenginering phase in which it is permanently modified. In the age of globalization changes need to be handled in a timely fashion. If the execution of the evolution loop is not able to quickly address changes, there is a risk of delivering a solution that addresses a certain change when this change is not relevant any longer, for instance, because it has been superseded by other new changes. Agile development approaches [7], [8] are often mentioned as a way to address changes in a fast and interactive way, if developers are able to work in close collaboration with the owners of new requirements or with those who have a deep knowledge about the occurred context changes. This statement has an initial evidence in the work of Capiluppi et al. [9] where authors have shown that agile methods have resulted in a ”smooth evolution while avoiding the problems of increasing complexity or decreasing customer satisfaction”. In the context of SOA, companies such as IBM [10] and OutSystems [11] are suggesting the adoption of an agile approach. Moreover, a technological advancement that pushes agile development of SOA to the extreme is offered by mashups [12].

Mashups constitute the integration of different web applications and services which have the purpose of serving the specific needs of some users. They are usually short lived systems intended to be built not only by expert developers but also by less-experienced people. To this end, proper development environments are being developed. These offer a specific component and composition models and are usually associated to some runtime environment. While these environments promise to shorten the development cycle in a significant way, they are still not well integrated into a proper full-fledged development methodology. From a completely different perspective, another interesting step toward agility is the integration of the service selection phase within requirement engineering [13].

Even though it has not been designed with agile processes in mind, agile methods could be mapped to the S-Cube lifecycle. Using Scrum [14] we could assume that each iteration of the evolution loop is performed as a Sprint which aims at delivering an increment of a service-oriented system. Given that services represent a natural unit of functionality, each such Sprint could aim at incorporating a new service in the service-oriented system up to the point where the complete functionality is offered to the system's users.

As the aim is to build self-adaptive systems, an agile approach targeted at service-oriented systems needs to take into account that the redesign and redevelopment of the software system has to incorporate the "design for adaptation" principle in a seamless way. This is an issue which surely deserves further research.

While the behavior of a non-adaptive system is only controlled by user input, adaptive systems consider additional information about the system (machine) and its context. Self-adaptive behavior is realized through methods and tools that realize control loops. In addition to the degree of automation, the point in time when a change can be detected impacts on the agility of the system in responding to that change; e.g., if the system can forecast an imminent change, more time and thus more options remain for the adaptation than if the system can only detect changes once they actually occur.

[3]http://aws.amazon.com

The following three types of adaptation [15] exemplify the impact that the point in time when changes are detected has on the agility of the system:

- *Reactive adaptation* refers to the case in which the system is modified in response to external failures that have actually occurred, i.e., failures that are actually observed by the users of the system. This approach is the most common one in the literature, but, being reactive, can have a severe impact on how agile a system can respond to changes [16].
- *Preventive adaptation* refers to the case in which an actual local failure or deviation is repaired *before* its consequences become visible to the user in the form of an external failure.
- *Proactive adaptation* refers to the case in which the system is modified even *before a local failure occurs*. In this case, neither repair nor compensation activities would be necessary as part of the adaptation, as no failure has actually occurred.

In a nutshell, the more agile the service-oriented system is to become, the stronger the role of proactiveness in the adaptation loop becomes. This means that already during design (i.e., within the evolution loop), decisions need to be made about how to predict failures during the execution of the service-oriented system. However, selecting the right technique can be quite a challenging task due to the highly dynamic nature of service-oriented systems (see [15] for an in-depth discussion on this issue).

V. Migration of Legacy Systems to Service-based Applications

Companies have extensive experience in both in-house migration of their own legacy information systems to a more agile, reusable service-oriented paradigm, and consultancy migration to support customer organizations to port their systems to modern service-oriented technologies, make them available as added-value services, often with the goal of creating new market opportunities. Researchers have investigated aspects of SOA migration for the past decade. While in SOA migration academics take a reverse engineering approach and propose methods, tools and techniques to port old systems to service-oriented ones, enterprises take a forward engineering approach, mostly by eliciting important knowledge about the requirements supported by the legacy software and re-engineering them into services that fulfill an ideal SOA [17], [18].

The question is: why this gap between SOA migration approaches by academic and practitioners exists? In our earlier work we have argued that this is due to the fact that academic approaches do not fit the fundamental problems, goals, strategies and weaknesses of practice [18]. Having identified the theory-practice gaps we now turn our attention to the future. Out of the results of a number of case studies, industrial surveys and panel with experts we found out that, in practice, there is a need for SOA migration strategies that are simple enough to be pragmatic and have very wide applicability. The best strategies are those that are based on a gradual migration approach. They have to be in-line with the mental models of practitioner, and eventually offer choices, possibilities and alternatives for important problems that are typical in SOA migration. We call such promising approaches as Lean & Mean Migration Strategies [19]. Considering this generic need, we believe the following represents major challenges for industry-relevant SOA migration approaches in the years ahead:

- *Aligning risks, costs and value with migration strategy:* One of the issues that was repeatedly stated by practitioners was the importance of risks and cost management in SOA migration decision making. We found risks and cost management to be in fact one of the main drivers of migration and influential on most decisions. This further confirms a recent interest toward risk-, cost- and value-aware methods [20], [21] that needs further research.
- *Providing decision making tools to support selection of migration solutions:* Industry needs tools to support planning and decision making for migration. For this purpose, the interviewed practitioners indicated as very beneficial to associate the practices or extensions with typical risks, costs and pre-requisites. This calls for empirical research studies to identify and isolate practices and associate them with important decision criteria such as risks, costs and pre-requisites.
- *To-Be-driven migration approaches:* Migration in industrial approaches migration is mainly driven by To-Be state (i.e., ideal target service-based system) [18]. Inadequate support for To-Be driven approaches in academia highlights promising opportunities for research. In particular, future research should focus on how to systematically elicit and capture the migration drivers and how to shape the migration process using those drivers.
- *Legacy understanding without reverse-engineering:* The industrial migration approaches do not use reverse engineering techniques to understand the legacy systems. The required knowledge is elicited from the stakeholders who own the knowledge. Elicitation of the knowledge about the legacy system, however, is crucial for a successful migration. In this regard, research can benefit practice by providing methods, techniques, or guidelines that facilitate elicitation of migration-relevant knowledge from different sources.

VI. Humans Providing Services

Major industry players have already been working on standardizing protocols and languages which allow people to interface with WS environments. The SOA community has recognized the emerging requirement of human skills and knowledge in service compositions. Some of the processes steps mapped to services still require support in complex decision making, and as a result, human expertise or knowledge to proceed. To address the lack of a possibility of human interactions in service-oriented businesses processes [22], specifications including BPEL4People [23] and WS-HumanTask [24] have been defined.

Our latest studies relate to environments with characteristics of open service environments involving humans providing their skills as a service.

Open service environments involving humans: Open Enterprise Systems (OES) are open collaborative networks organized in communities. Communities are established by members with the same interests and skills. This information is provided by the members' profile. A typical scenario includes a requester that requires human assistance with an activity. The profiles help to find the matching expert. The requester selects the expert, submits the activity and awaits the response. Examples of OES employing SOA infrastructures have been studied in [25], [26]. These include small and medium-sized companies and their bilateral alliances to compete with global players. This protects the partners against the dynamics of economy and business, supporting them also to harvest business opportunities that a single partner cannot take. The result of these associations is referred to as *virtual organizations* supporting enterprise collaboration. Particular instances have been explored in [27]. The example outlines a science collaboration network. It comprises scientists, members from national and international research labs, and experts from the industry. Furthermore, professional virtual communities are discussed in [28] that provide help and support on requests of each other, e.g., law firms and insurance companies.

Crowdsourcing Environments. The recent trend towards *collective intelligence* and crowdsourcing can be observed by looking at the success of various Web-based production platforms that have attracted a huge number of users. Crowdsourcing applications [29] are online, distributed problem-solving and production models that have emerged in recent years. They are typically open Internet based platforms where problem-solving tasks are distributed among a group of humans. Crowdsourcing follows the *open world assumption* and generally comprises three roles identified in [30]. First, there is the crowdsourcing platform and its owner, also referred to as the platform *provider*. A crowd *customer* is interested in outsourcing tasks to the crowd. The registered vast number of crowd *workers* with different skills offers then individual solutions to the outsourced problem. Apart from its benefits of multiple redundant workforce and collective intelligence, many of the challenges of crowdsourcing are related to its distributed and open nature. The main challenges remain how to organize and manage the crowd and identify potentially missing skills [29]. In the following different types of crowdsourcing environments are described.

Alignment with Mixed Systems: A possible integration of the aforementioned environments into an SOA infrastructure includes the roles of the *service consumer*. The consumer usually considers outsourcing an activity to the environment because of the lack of in-house skills related to the activity. Considering the environment itself as based on a WS technology, the mixture of services includes two different types. The *Human-provided Services (HPS)* are the services which represent interaction humans in the service system and the *Software-based Service (SBS)* are the traditional WS. In some of our previous works [31], [32] we took an approach from the environment provider's point of view. As our results show, with their standardized formats for communication, WS allow monitoring and analyzing the interactions in a straightforward manner. One possible application which can be regarded as a starting point for several future evaluations is the extraction of *areas of interest* and *social relations*.

VII. Applying a Service Model in Global Software Engineering

Global Software Engineering (GSE) involves a complex interplay of globally distributed virtual teams. Space, time and social distances typical of the GSE condition, create problems in teams collaboration (which is social by definition). To support GSE, mechanisms must be provided to model and sustain its Organizational Social Structure (OSS), i.e. the web of relations, collaborations and social ties that emerges among GSE developers [33]. These mechanisms must address social and collaboration problems (e.g. mutual awareness of virtual teams involved).

We argue that supporting the GSE OSS is possible by modeling a GSE process as an *Agile Service Network* (ASN), that is, a network of services (nodes) collaborating through adaptive transactions (edges), towards a common goal [34]. This can support the GSE condition, for instance, through a project-specific social network to ease the localization of needed skills, their communication, or seamless end-to-end information propagation across timezones.

A. Sample Scenario

Consider the following wicked GSE "social mess" (obtained by enhancing a scenario in [35]): An unavoidable technical problem is discovered by two programmer teams in different timezones. This causes a multiple problem "vision". Designers (alerted twice for the same issue) rework design with two subsequent (and conflicting) solutions. Negotiation and adjustment of requirements are only partial (e.g. related to the first re-design), since system requirements analysts were only alerted of the first rework as the second rework seemed only minor. In addition, alert propagates only to teams which are alive (i.e. are working) at the time the technical constraint is discovered. Subsequently, the same technical problem is raised by other programmers who were using design details unrelated to the adjusted design, but related to affected requirements.

B. Proposed Solution

Given that we see a GSE team as an ASN, we propose to address the above case and, in general, the coordination issues arising in a GSE team, through the development of a tool that we call *S.E.A.N.* (Socially-Enabled Awareness Network). *S.E.A.N.*'s purpose is to maintain the mutual awareness of the GSE crowd constant, so that collaborative workers are kept mutually aware of their progress (or difficulties, both personally encountered or reported by other neighbors). Using basic network theory, *S.E.A.N.* can be described by (semi-)formalizing its edges and nodes:

- Each node is a service-based social networking application, that represents a single developer X, working on project Y.
- Each edge represents the collaboration between two developers on the same task of project Y.

S.E.A.N. uses collaborator discovery services to assemble a network of service applications which emerges spontaneously based on who works on which project, as well as on same (or related) tasks. It is business-oriented since its prime goal is to minimize development time and wasted effort (business gain) by increasing awareness of the developers. It is decentralized since no single node controls the operational effectiveness of the network. It is collaborative, since each node effectively works on increasing the project awareness that every other node perceives. It is dynamic since it deploys basic context awareness and adaptation mechanisms (e.g. the fail-safe messaging service).

To model a *S.E.A.N.* instance of the above scenario, let's assume it entails 6 teams in total: 2 programmer teams (initially discovering the technical constraint); 1 designer team (reworking the design); 1 requirements team (adjusting and maintaining requirements) and 2 tester teams (re-discovering the technical constraint after their working shift starts).

Text-book software engineering helps in bootstrapping the (loosely-)bound collaborations between teams: for example, the two programmer teams constitute a collaboration mesh, since they work on a common codebase; also, leaders in the programmer teams collaborate to design adjustments, to limit excessive clashes between the design and the current codebase. Designers also collaborate on requirements engineering with analysts since requirements are bound to design elements (and decisions). Finally, testers should mesh with the rest of the network since the whole project should reflect the adjustments they dictate.

In *S.E.A.N.*, all the developers in the engineering effort are subscribed to a project which was created with *S.E.A.N.*. A single, high priority message is risen when a design needs rework and transmitted to all active collaborations from the programmer team (and to the neighbor's neighbors). Designers are immediately notified (once). The designers work out the flaw, helped by other designers' contributions in other nodes. A single solution is applied to the design and transmitted to the requirements. A high priority message is pushed seamlessly when the risen issue is closed.

This intuitive validation however only exemplifies our argument. Prototyping and action research should be used jointly to concretely assess this idea.

VIII. Conclusion

Service-based applications show interesting characteristics that open up a large number of challenges in software engineering. Some of these challenges concern specifically the approaches and ingredients to be used to support their development. Other challenges concern the capability of service-based applications to incorporate and model the behavior of humans and team of humans. These last aspects have not been deeply investigated in the literature yet, and constitute an interesting area for future investigation.

Acknowledgments

Research leading to these results has received funding from the European Community's 7th Framework Programme FP7 / 2007-2013 under grant agreement 215483 (S-Cube). This research received funding from project Jacquard SAPI-ENSA (contract 638.001.206).

References

[1] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engineering*, 2008.

[2] M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, Eds., *Service Research Challenges and Solutions for the Future Internet: Towards Mechanisms and Methods for Engineering, Managing, and Adapting Service-Based Systems*. Heidelberg, Germany: Springer, 2010.

[3] R. de Lemos et al, "Software Engineering for Self-Adpaptive Systems: A second Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, ser. Dagstuhl Seminar Proceedings, R. de Lemos, H. Giese, H. Müller, and M. Shaw, Eds., no. 10431. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2011.

[4] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 2, 2009.

[5] D. Meiländer, A. Bucchiarone, C. Cappiello, E. D. Nitto, and S. Gorlatch, "Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds," vol. 7221. Springer, 2011, to appear.

[6] A. Ploss, D. Meiländer, F. Glinka, and S. Gorlatch, *Towards the Scalability of Real-Time Online Interactive Applications on Multiple Servers and Clouds*. IOS Press, 2011, to appear.

[7] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New directions on agile methods: a comparative analysis," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 244–254.

[8] D. Cohen, M. Lindvall, and P. Costa, “An introduction to agile methods,” *Advances in Computers*, pp. 1–66, 2004.

[9] A. Capiluppi, J. Fernandez-Ramil, J. Higman, H. Sharp, and N. Smith, “An empirical study of the evolution of an agile-developed software system,” in *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, may 2007, pp. 511–518.

[10] P. Krogdahl, G. Luef, and C. Steindl, “Service-oriented agility: An initial analysis for the use of agile methods for SOA development,” in *IEEE International Conference on Services Computing*. Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 93–100.

[11] D. Sprott, “Product Overview: OutSystems Agile SOA Platform,” *CBDI Journal*, pp. 15–20, April 2009.

[12] J. Yu, B. Benatallah, F. Casati, and F. Daniel, “Understanding mashup development,” *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, 2008.

[13] K. Zachos and N. Maiden, “Inventing requirements from software: An empirical investigation with web services,” in *Proceedings 16th IEEE International Conference on Requirements Engineering*. IEEE Computer Society Press, 2008, pp. 145–154.

[14] K. Schwaber, *Agile project management with Scrum*. Microsoft Press Redmond (Washington), 2004, vol. 7.

[15] A. Metzger, “Towards accurate failure prediction for the proactive adaptation of service-oriented systems (invited paper),” in *Proceedings Workshop on Assurances for Self-Adaptive Systems (ASAS), collocated with ESEC 2011*, 2011.

[16] J. Hielscher, R. Kazhamiakin, A. Metzger, and M. Pistore, “A framework for proactive self-adaptation of service-based applications based on online testing,” in *ServiceWave 2008*, ser. LNCS, no. 5377. Springer, 10-13 December 2008.

[17] M. Razavian and P. Lago, “A Frame of Reference for SOA Migration,” in *Towards a Service-Based Internet*, ser. Lecture Notes in Computer Science, E. D. Nitto and R. Yahyapour, Eds., vol. 6481, 2010, pp. 150–162.

[18] M. Razavian and P. Lago, “A Survey of SOA Migration in Industry,” in *International Conference on Service Oriented Computing, ICSOC*, G. Kappel, Z. Maamar, and H. R. Motahari-Nezhad, Eds., 2011.

[19] M. Razavian and P. Lago, “The How and Why of SOA Migration in Industry,” under submission, 2012.

[20] E. R. Poort and H. van Vliet, “Architecting as a risk- and cost management discipline,” in *Ninth Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2011.

[21] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, and N. Zazworka, “Managing technical debt in software-reliant systems,” in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 47–52.

[22] F. Leymann, “Workflow-based coordination and cooperation in a service world,” in *CoopIS, DOA, GADA, and ODBASE*, 2006, pp. 2–16.

[23] M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, and I. Trickovic, “Ws-bpel extension for people–bpel4people,” *Joint white paper, IBM and SAP*, 2005.

[24] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau *et al.*, “Web services human task (ws-humantask), version 1.0,” *June, available at http://download. boulder. ibm. com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask v1. pdf*, 2007.

[25] L. Camarinha-Matos and H. Afsarmanesh, “Collaborative networks,” in *PROLAMAT*, 2006, pp. 26–40.

[26] N. Schuster, C. Zirpins, and S. Tai, “A document-centric approach to open collaboration processes,” *Current Trends in Web Engineering*, pp. 538–544, 2010.

[27] F. Skopik, D. Schall, and S. Dustdar, “Trust-based adaptation in complex service-oriented systems,” in *ICECCS*. IEEE, 2010, pp. 31–40.

[28] F. Skopik, D. Schall, and S. Dustdar, “Trustworthy interaction balancing in mixed service-oriented systems,” in *SAC*. ACM, 2010, pp. 799–806.

[29] D. C. Brabham, “Crowdsourcing as a model for problem solving: An introduction and cases,” *Convergence*, vol. 14, no. 1, pp. 75–90, 2008.

[30] M. Vukovic, “Crowdsourcing for Enterprises,” in *Proceedings of the 2009 Congress on Services*. IEEE Computer Society, 2009, pp. 686–692.

[31] H. Psaier, L. Juszczyk, F. Skopik, D. Schall, and S. Dustdar, “Runtime behavior monitoring and self-adaptation in service-oriented systems,” in *Self-Adaptive and Self-Organizing Systems (SASO), 2010 4th IEEE International Conference on*. IEEE, 2010, pp. 164–173.

[32] H. Psaier, F. Skopik, D. Schall, and S. Dustdar, “Resource and agreement management in dynamic crowdcomputing environments,” in *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*. IEEE, 2011, pp. 193–202.

[33] E. C. Wenger and W. M. Snyder, “Communities of practice: The organizational frontier,” *Harvard Business Review*, vol. 78, no. 1, pp. 139–+, Jan. 2000.

[34] D. A. Tamburri and P. Lago, “Supporting communication and cooperation in global software development with agile service networks,” in *ECSA*, 2011, pp. 236–243.

[35] N. S. Shami, N. Bos, Z. Wright, S. Hoch, K. Y. Kuan, J. S. Olson, and G. M. Olson, “An experimental simulation of multi-site software development.” in *CASCON*, H. Lutfiyya, J. Singer, and D. A. Stewart, Eds. IBM, 2004, pp. 255–266. [Online].