All the services large and micro: Revisiting industrial practices in services computing

Services computing is both, an academic field of study looking back at close to 15 years of fundamental research, as well as a vibrant area of industrial software engineering. Industrial practice in this area is notorious for its ever-changing nature, with the state of the art changing almost on a yearly basis based on the ebb and flow of various hypes and trends. In this paper, we provide a look "across the wall" into industrial services computing. We conducted an empirical study based on the service ecosystem of 42 companies, and report, among other aspects, how service-to-service communication is implemented, how service discovery works in practice, what Quality-of-Service metrics practitioners are most interested in, and how services are deployed and hosted. We argue that not all assumptions that are typical in academic papers in the field are justified based on industrial practice, and conclude the paper with recommendations for future research that is more aligned with the services industry.

All the Services Large and Micro: Revisiting Industrial Practice in Services Computing

Gerald Schermann, Jürgen Cito, and Philipp Leitner

University of Zurich, Switzerland
{schermann, cito, leitner}@ifi.uzh.ch

Abstract. Services computing is both, an academic field of study looking back at close to 15 years of fundamental research, as well as a vibrant area of industrial software engineering. Industrial practice in this area is notorious for its ever-changing nature, with the state of the art changing almost on a yearly basis based on the ebb and flow of various hypes and trends. In this paper, we provide a look "across the wall" into industrial services computing. We conducted an empirical study based on the service ecosystem of 42 companies, and report, among other aspects, how service-to-service communication is implemented, how service discovery works in practice, what Quality-of-Service metrics practitioners are most interested in, and how services are deployed and hosted. We argue that not all assumptions that are typical in academic papers in the field are justified based on industrial practice, and conclude the paper with recommendations for future research that is more aligned with the services industry.

1 Introduction

Since the inception of standardised, XML-based service definition, description and discovery languages and approaches [6] (i.e., the WS-* stack) around the year 2002, academic research has zealously embraced the ideas of service-oriented computing and Service-Oriented Architecture (SOA) to build and organize large-scale distributed applications. However, services computing is not a static field. Over the years, various new industry-driven technological trends (e.g., REST [15], enterprise service buses or ESBs [16], cloud computing [4], or most recently microservices [13]) have appeared, and became integrated into how academic researchers think about services. The disadvantage of this integrative approach is that, by now, the term "service-based application" (SBA) can mean any number of things, ranging from dynamic SOAP- and WSDL-based applications built using the traditional triangle of publish-find-bind [11], WS-BPEL-based compositions of public Web services, large-scale, heterogenious, enterprise services connected via an ESB, all the way to microservices-based cloud applications (or any combination thereof).

Orthogonally, but relatedly, a sometimes voiced criticism of current academic services research is that it is too removed from industrial practice. To give just one example, Prof. Anthony Finkelstein (University College London) has in a blog entry remarked that research in service discovery deals with a problem that very few practitioners actually have[1]. However, non-anecdotal data about the practical impact and relevance of services research is hard to come by.

In this paper, we aim to provide the academic community a glance "across the wall" into industrial services computing. We conducted a small-scale survey of the state of practice in services computing, with the primary goal of understanding what practitioners mean when they talk about services, how they technically implement and host services, and what issues they struggle with. Our study has been set up with a specific focus on the recent trend of microservices. We hope to contribute to services research by painting a clearer picture of how service-based applications actually look like in practice, which issues require better approaches, and which traditional research areas in the field are simply not all that relevant in practice. Note that we focus specifically on *technical* issues of services computing in this paper. To keep the size of the research managable, we excluded economic and cultural topics in this study. Further, we limit our research to technical services, and exclude questions on human-provided services, workflows and business processes.

It should be noted that the goal of this workshop paper is to provide a starting point for fruitful discussion, not to critize individual researchers or the community at large. The third author of this paper has himself published on all individual research ideas that are going to be put into question in the following. Further, given that our sample size is not overly large, we do not claim to have all the answers. There is certainly potential for more large-scale and more rigorous follow-up research. We primarily follow an empirical approach. Using a Web-based survey, we questioned 42 companies with one or more service-based products. Our results show that most service ecosystems are of quite managable size. While public Web services are in use, most services are actually internally developed and operated. REST and HTTP are almost ubiquitious, while SOAP is falling out of favor fast. Most companies do not make use of a centeralized composition engine or service bus. Instead, most service interactions follow what researchers would call a service choreography style. Cloud computing and QoS monitoring is indeed of large industrial relevance today.

2 Study Setup and Method

We conducted our research as a quantitative, Web-based survey. We targeted developers and companies that self-identify as building a service-based product or making use of a service-oriented architecture. To acquire participants, we advertised our study on multiple programming-related Web sites, as well as through personal contacts and social media. We were able to acquire 42 participants, which were close to equally distributed over companies of all sizes, ranging from 1 - 20 employees up to global enterprises with more than 1000 employees. About

[1] `http://blog.prof.so/2012/06/bottom-10-software-engineering.html`

50% of our study participants are working as software developers. The bulk of the remaining participants where either team leads, DevOps engineers, or product owners. Most of our participants are experienced software developers, with close to three quarters reporting seven years of experience or more. This data is summarized in Figure 1.

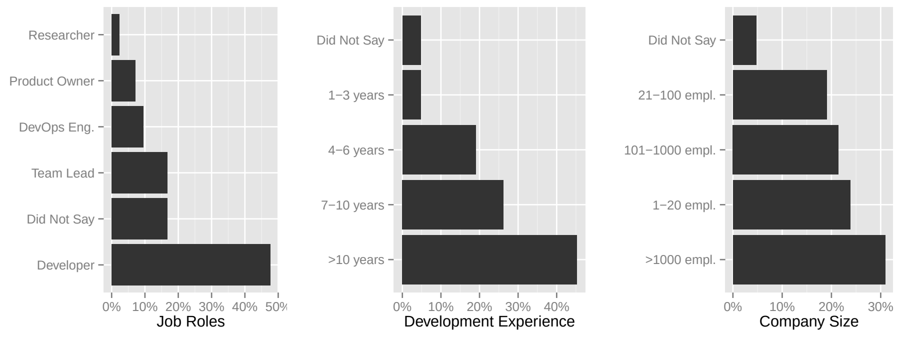

Fig. 1. Demography of study participants. Most participants are experienced and work as software developers in large enterprises.

Our study consisted of 25 questions, designed as either multiple choice, single choice, or open-ended free text questions. When designing our study we strived for a good compromise between keeping the study short for the participants and collecting material related to a wide range of currently "hot" topics in academic services research. Finally, and after discussions with our industrial partners and internal testing of the survey, we decided to ask questions about technical fundamentals, middleware, service discovery, QoS monitoring, and cloud deployment, leading to a Web-based survey that took our participants less than 10 minutes in the median to complete. A complete list of questions, as well as all resulting data, is available as part of the online appendix.

3 The State of Practice in Service-Based Applications

We now discuss the outcomes of our research. For reasons of brevity, we only summarize the most important outcomes. All raw data is available as part of our online appendix.

3.1 Fundamentals

First, we discuss the technical fundamentals of services. What is the typical size and complexity of services, to what extent are external services used, and what are the common programming languages for developing services?

Services vary in size, but few are truly "micro". An evergreen question in services computing is the "optimal" size of individual services. Erl refers to services as "coarse-grained entities" [7], implying that each service should carry substantial business logics. The current microservices trend emphasizes tiny services, with "10 to 100 lines of code" (LOC) each[2]. We asked our participants about the typical size of services within their organization in LOC. We observed that services in the range of 1'000 to 10'000 LOC are dominant, as stated by 51% of our participants, followed by 100 to 1'000 LOC chosen by 43%. Services following the microservices rule of thumb (less than 100 LOC) are rare (3%), as are very large services with more than 10'000 LOC.

Services are dedicated. Another interesting question is how many separate concerns an individual service covers. Following most literature, services are supposed to be dedicated to a single task. As a metric to measure this, we asked our participants how many public operations a service typically provides. The resulting data shows that services seem to indeed typically be dedicated to relatively narrow tasks, exposing between 1 to 9 (46%) or 10 to 20 (45%) public operations. The remaining 19% of our participants operate services with relatively large public interfaces (between 20 and 50 operations).

Most service ecosystems are actually not very large. Much academic research in services computing is motivated by a presumed large number of services to compose applications from. To this end, we have asked our participants how many services they actually have access to within their organization, including in-house and usable external services. Our respondents stated to, in the median, only have access to 30 services. However, the individual answers to this question varied enourmously in a range of 7 to 20'000 services. This is because the background of the study participants also varied. Clearly, developers in globally operating enterprises typically have access to substantially more services than startup employees. However, only 25% of all participants actually deal with service ecosystems of substantial size (more than 100 services) on a regular basis.

Most companies use external services. A similar common assumption in academic research is that companies often make use of external services to implement their business goals. According to our responses, almost two thirds (64%) of our participants indeed make use of external services. However, 68% of all services that they use are actually internally developed. That is, most companies use external services, but the majority of services in use are still developed and operated internally.

Java is still the most common way to implement services. To conclude technical fundamentals, we were interested in how services are actually developed. The microservices trend often emphasizes a heterogenity of programming languages within an organization ("the right tool for the job"). This was not confirmed in our research. Indeed, we found that 45% of respondents use at most two programming languages, followed by 25% using three or four and 20% using five or six programming languages. The remaining 10% seem to have

[2] `http://guidesmiths.com/blog/the-granularity-of-a-micro-service/`

a strongly heterogeneous service architecture, as they implement their services in more than 6 different programming languages.

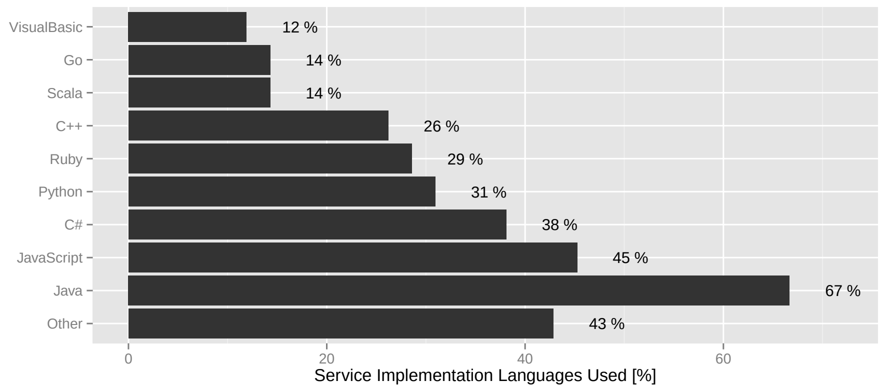

Fig. 2. Java is used to implement most services, followed by JavaScript and C#. Multiple selections were possible.

Further, we asked our participants what concrete programming languages they use for developing their services. As illustrated by Figure 2, Java was selected by 67% of our participants. Besides Java, we identified a strong focus on scripting languages, such as JavaScript, Python, and Ruby, excelling prominent compiled languages such as C++, C#, and Visual Basic. Finally, it should be noted that WS-BPEL or any other service composition language has not been mentioned as a typical service implementation language by any participant.

Key Points. Services vary substantially in size, but true "micro" services are rare. Many companies use some external services, but most services in use are internal. Java and JavaScript are the most common service implementation languages today.

3.2 Communication between Services

In this section, we focus on how services communicate with each other. Are services more commonly implemented using SOAP/WSDL, or is REST by now more relevant?

HTTP and REST are ubiquitous. The selection of a communication protocol or technology strongly depends on whether synchronous or asynchronous service-to-service communication is preferred. Almost all participants (95%) operate synchronous services that are based on HTTP and REST. The dominance of HTTP and REST fits well with the characteristic of microservices being built on top of lightweight communication mechanisms. However, services that communicate via message queues (e.g., AMQP), an asynchronous, event-based communication style, are also used by 57% of our participants. Interestingly, every participant that stated to use message queues for service-to-service communciation also operates at least one REST service. Only 21% still rely on RPC-based communication technologies, such as RMI or XML-RPC.

JSON is more common than plain XML, SOAP is less common than either. In terms of data exchange formats, we have seen that JSON has widely superseded XML as the primary service data exchange format. As summarized in Figure 3, 90% of our participants stated to use JSON as data exchange format, while plain XML was the choice of 57%. This can partially be explained with the increasing importance of JavaScript-based frameworks, such as Node.js. SOAP is not overly wide-spread in our study (40%). Google's Protobuf is on the rise, but still relatively rare with 17% usage across participants.

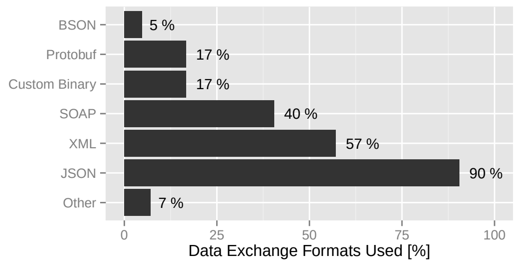

Fig. 3. JSON has replaced XML as primary data exchange format. SOAP is not overly common. Multiple selections were possible.

Dedicated service middleware is not often used. Finally, our study has shown that centralized, heavy-weight middleware (e.g., ESBs or composition engines) are not overly common. Rather, 45% are not using any middleware at all, which is an interesting fact and highlights the tendency to a more decentralized, choreographed approach rather than a central orchestration point. 31% of our participants use ESB technology for communication between services. The vast majority of those participants are employed at companies with 100 employees or more. API gateways (e.g., Swagger, Tyk, or Strongloop) are only in use at 19% of our respondents.

Key Points. HTTP and REST are used by 95% of our participants' companies. JSON has replaced XML as the most common data exchange format. Half of our of participants does not use any middleware at all, there is a trend towards a more choreography-style of managing service coordination.

3.3 Monitoring and Quality-of-Service

Measuring and monitoring Quality-of-Service (QoS) [12] has historically been considered an important and valuable field of research, but to what extent do practitioners care about QoS?

Standard QoS attributes are indeed widely monitored and used. As indicated in Figure 4, our study participants indeed monitor and use a broad spectrum of infrastructure (e.g., CPU utilization, network traffic) and application metrics (e.g., response times, failure rates). However, those metrics are rarely used to select services. Rather, metrics are used at runtime to monitor the health and performance of services. Our study participants use a wide range of monitoring tools, the most common of which are Logstash, Nagios, and NewRelic.

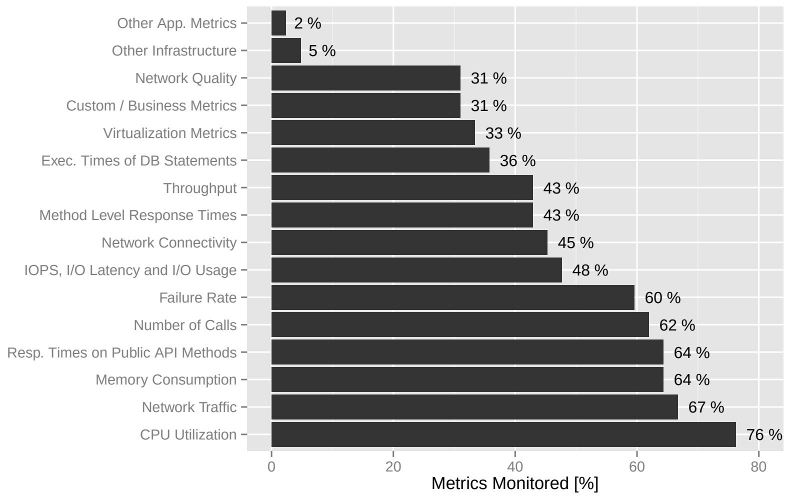

Fig. 4. Participants reported a wide range of QoS metrics being monitored, including system-level and application-level metrics. Business metrics are only used by 31% of all participants. Multiple selections were possible.

Business metrics are rare. A more interesting result is that only 31% of participants monitor any custom or business metrics. Given that various microservices proponents regularly emphasize the importance of business metrics as basis for development and business decisions, this number was behind our expectations. It seems that for most companies, standard performance metrics are sufficient today.

Key Points. Companies monitor a wide range of standard metrics on application and infrastructure level. These metrics are used to observe the health state of the application rather than to select services. Business metrics are used less than expected.

3.4 Service Discovery

Assuming a service ecosystem with hundreds or thousands of services, discovering the right service is challenging. How do companies handle service discovery in practice, do they use service registries and how do they find out how to invoke services?

Registries are not commonly used in practice. Even though actively researched in the previous 15 years, service registry and discovery concepts such as UDDI have never gained much attention in industrial practice. This is also reflected in our participants' responses when we asked them how they know if certain functionality is available as a service. Only 18% stated that they have a middleware for registering and querying services. 28% of participants are manually maintaining a list, website, or WIKI page of available services. 25% stated that service discovery is a minor issue as they do not have that many services and just know what is available. Similarly, 18% mentioned that there is a contact person within the organization to ask about what services are available.

Client-side dynamic binding of services is not typically used. We have not seen a strong indiciation that practitioners actually follow the "SOA triangle" of publish-find-bind in any real way. 70% of our participants rely on a documented fixed configuration which does not change, or use server-side approaches (e.g., DNS) to manage service binding. Our participants generally do not make use of client-side dynamic binding approaches, such as QoS-aware service selection.

Key Points. 28% of our participants manually maintains a list of available service functionality. Documented, fixed configurations are the most common way to bind clients to services.

3.5 Service Hosting and Deployment

Finally, we were interested in how services in the wild are actually hosted and deployed. The academic literature has widely embraced the notion of cloud computing as means to house and provision services, but to what extent does this reflect industrial practice?

Cloud computing is mainstream. Our study results, which are also depicted in Figure 5, indicate that by now cloud computing has indeed found its way into the mainstream of industrial services computing. Two thirds of all participants use either public or private cloud systems to host their applications. Only 50% of all participants even still have services that are hosted in-house on non-virtualized infrastructure. A third of our respondents are still using long-term external hosting providers. Despite cloud computing being sometimes branded as primarily interesting to startup companies [9], there is no significant difference between the data of small companies and large enterprises in our study.

Interestingly, even though recent study results indicate that elasticity and automated scaling are primary drivers in cloud adoption [5], most services (70%) are currently actually *not* scaled automatically. However three quarters of participants use the cloud to redundantly deploy their services, mostly to improve fault tolerance. Half of our respondents even deploy their cloud services redundantly over three or more nodes.

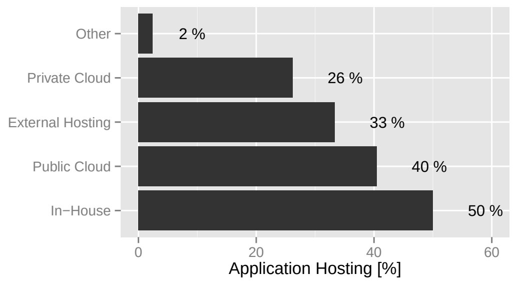

Fig. 5. 66% of all participants use public or private cloud services. Only 50% still use in-house hosting. Multiple selections were possible.

Application packages are still widely used. Another interesting outcome of our research is that building and copying application archive packages (e.g., Java JAR or WAR files) is still the by far most common way to provision the implementation code of cloud services, presumably as part of a Continuous Integration toolchain (e.g., Jenkins). Container technologies (e.g., Docker or LXC) are on the rise, but currently only in use at 34% of all participants. 21% use virtual machine formats (e.g, Amazon Machine Images or VMWare Images). Small minorities of respondents used UNIX packaging mechanisms or provisioned code directly out of the version control system onto cloud instances (e.g., they clone the code from Git as part of provisioning).

Key Points. Cloud computing is already widely adopted in practice. However, most cloud services are not scaled automatically. Provisioning is still mostly done by building and deploying application packages as part of a Continuous Integration toolchain.

4 Recommendations for Research

The main goal of our study was to survey the state of practice to guide future services computing research. Hence, we now (somewhat provocatively) discuss some implications of our results for a number of common research themes in the field.

***Do not* assume that service ecosystems are huge.** Many academic works on service selection are motivated by a presumed humongous number of services to choose from. This is typically not the case, except within a few international corporations. Most service ecosystems are quite easy to track even manually (e.g., via WIKIs). One aspect of this is also that public, external Web services are not quite as prevalent as some research works seem to assume.

***Do not* assume that there are many alternative services to choose from.** In our study we have not seen any particular indication that practitioners indeed commonly need to choose from a list of functionally comparable services. As most services are internal, there is typically exactly one (in some cases two, including a legacy system) service that implements any particular business need. In light of this, academics should reflect whether more attention to approaches for client-side dynamic binding and dynamic service selection is warranted.

***Do not* assume that Web services always use SOAP.** Our results have shown that SOAP is certainly not a de-facto standard in the Web services field anymore. If a research work needs to assume a specific service style, it should probably be REST and HTTP rather than SOAP. Consequently, the importance of WSDL should also be reconsidered.

***Do* research on choreography rather than orchestration.** Our participants largely do not make use of centralized composition engines or service buses. Particularly in smaller service ecosystems, services are composed in an ad hoc, decentralized, choreography style. We argue that these kinds of service compositions deserve more research attention, particularly in the light of the current microservices trend.

***Do* research on QoS, but for monitoring rather than service selection.** Our results show that QoS is indeed a "hot topic" in practice. Even though the state of practice in this area is quite mature, it is our impression that there are still interesting research questions to be addressed. However, academics should not assume that QoS is primarily used as a distinguishing factor between functionally comparable services.

***Do* research on cloud computing, but *do not* assume that every cloud-deployed service is elastic.** Cloud computing is indeed often used in practice, and we argue that the current research attention is warranted. However, there seems to be a trend among current research works to equate cloud computing with elasticity. Our results have shown that there are many, heterogenious reasons why practitioners use the cloud. Academics should not assume that every service deployed to the cloud is necessarily elastic.

5 Related Work

Quantitative empirical research methodologies, such as the one used in our study, are not overly common in the services field. A small number of empirical studies are available, but those are typically focused on a single product (e.g., IBM Jazz [2], SAP [1]) or domain (e.g., telecommunications [8], the financial industry [10]). While many publications present (more or less sophisticated) case studies (e.g., [17–19]), we are not aware of any recent academic publication that systematically validated some of the long-standing assumptions of the research area on a larger and more heterogenious sample of practitioners. Consequently, the research roadmaps of the field (e.g., [14]), as well as reference architectures (e.g., [3]), have historically been driven primarily by academic interests and opinions rather than quantified industrial needs. We argue that this has led to a positive feedback loop for some topics, where many published papers on the topic signified relevance to academics, leading to even more papers on the topic being published, despite little actual industry uptake. It is our hope that our research can serve as a useful tool that allows researchers to reality-check whether their assumptions are plausible for industrial practice. However, ultimately, more and more rigorous empirical data will be necessary to move services computing forward.

6 Conclusion

Our goal in this study was to provide a peek into the current state of practice in services computing. We have surveyed 42 practitioners working in companies of widely varying size. Our results indicate that most service ecosystems are small and consist mostly of internal services. The REST paradigm is very popular. Service choreography is more commonly used than central orchestrators. Cloud computing is of large industrial relevance, but not everybody who uses the cloud does so because of elasticity.

Our goal with this paper was primarily to motivate researchers working on services computing to reflect on the practical relevance of their work, and to occasionally revisit long-standing and often-repeated assumptions. We argue that the services computing field would benefit from more empirical studies being conducted to ground the basic research. Due to the small sample size and large scope, our work can only serve as a first step into this direction.

Online Appendix

The survey and resulting data that this paper is based on is available online[3].

References

1. Rama Akkiraju and Anca Ivan. Discovering business process similarities: An empirical study with sap best practice business processes. In Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *ICSOC*, volume 6470 of *Lecture Notes in Computer Science*, pages 515–526, 2010.
2. Laura Anderson, Bala Jegadeesan, Kenneth Johns, Mario Lichtsinn, Priti Mullan, James Rhodes, Akhilesh Sharma, Ray Strong, and Ruoyi Zhou. Enhancing Collaboration with IBM's Rational Jazz^{tm}. In *Service-Oriented Computing - 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010. Proceedings*, pages 501–514, 2010.
3. A. Arsanjani, Liang-Jie Zhang, M. Ellis, A. Allam, and K. Channabasavaiah. S3: A service-oriented reference architecture. *IT Professional*, 9(3):10–17, May 2007.
4. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616, June 2009.

[3] `http://wp.ifi.uzh.ch/leitner/?p=743`

5. Jürgen Cito, Philipp Leitner, Thomas Fritz, and Harald C. Gall. The Making of Cloud Applications An Empirical Study on Software Development for the Cloud. In *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE)*, New York, NY, USA, 2015. ACM.
6. Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the web services web: An introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, 6(2):86–93, March 2002.
7. Thomas Erl. *Service-oriented architecture (SOA): concepts, technology, and design.* Prentice Hall, 2005.
8. D. Griffin and D. Pesch. A survey on web services in telecommunications. *Communications Magazine, IEEE*, 45(7):28–35, July 2007.
9. Prashant Gupta, A. Seetharaman, and John Rudolph Raj. The Usage and Adoption of Cloud Computing by Small and Medium Businesses. *International Journal of Information Management*, 33(5), 2013.
10. James Lawler, Zheng Li, Nasir Javed, Dennis Anderson, Jonathan Hill, and Hortense Howell-Barber. A study of web services projects in the financial services industry. *IS Management*, 22(1):66–76, 2005.
11. P. Leitner, F. Rosenberg, and S. Dustdar. Daios: Efficient dynamic web service invocation. *Internet Computing, IEEE*, 13(3):72–80, May 2009.
12. Daniel A. Menascé. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, November 2002.
13. Sam Newman. *Building Microservices.* O'Reilly, 2015.
14. Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38–45, November 2007.
15. Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big"' web services: Making the right architectural decision. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.
16. M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The enterprise service bus: Making service-oriented architecture real. *IBM Syst. J.*, 44(4):781–797, October 2005.
17. Sebastian Stein, Stefan Kühne, Jens Drawehn, Sven Feja, and Werner Rotzoll. Evaluation of orvia framework for model-driven soa implementations: An industrial case study. In Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors, *Business Process Management: 6th International Conference*, pages 310–325. Springer, LNCS 5240, Milan, Italy, September 2008.
18. W.T. Tsai, X. Wei, Y. Chen, B. Xiao, R. Paul, and H. Huang. Developing and assuring trustworthy web services. In *Autonomous Decentralized Systems, 2005. ISADS 2005. Proceedings*, pages 43–50, April 2005.
19. W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. Alves de Medeiros, M. Song, and H. M. W. Verbeek. Business process mining: An industrial application. *Inf. Syst.*, 32(5):713–732, July 2007.