Towards Autonomic Processes and Services

Schahram Dustdar

Distributed Systems Group
Institute of Information Systems
Vienna University of Technology,
Vienna, Austria
{dustdar@infosys.tuwien.ac.at}

Abstract. More than ever, computing devices are becoming more powerful and networked, organizational boundaries are dissolving, and underlying information systems become more complex, thus requiring higher degrees of autonomic behavior of the business processes and software services they support. In this keynote talk the main challenges towards building the required novel conceptual abstractions as well as needed technological implementations are presented and discussed.

Keywords: Service-oriented Computing, Web services, Autonomic Computing, Service Composition, Context-based services

1 Introduction

In the past several decades the industrial landscape changed dramatically. Novel business models were increasingly introduced and successfully implemented. More recently, the vision of Service-oriented Architecture (SOA) aims at providing a model to allow realization of such novel, highly dynamic, adaptive, and composeable information systems and services for such business models and processes. SOAs are, in fact, mapping the real world unto the world of large-scale Internet-based information systems. Today we find many businesses and industries being "service-oriented". For example, telecommunications, financial services, healthcare, logistics, just to name a few. Those industries became "service-oriented" mainly through three factors: specialization, standardization, and scalability. All those factors can be also witnessed as being crucial in our educational systems. Standardization, in particular, is an important factor in the world of SOAs and business processes. In fact, it seems that – as we see in the real world in many examples (e.g., Starbucks) we increasingly move to global standards of various products and services. In the Internet-world the same principle is applied to SOAs: Standards are being agreed upon and introduced (e.g., the Web services stack) and novel methods for building such global large-scale systems are being promoted:

The SOA for the top-down enterprise-scale approach to business process design and service composition (build once and use many times), and more recently, the service mashup approach (build once and use once), for the bottom-up end-user (consumer) driven approach to service composition. Service mashups have some additional characteristics, such as more or less concurrent design and execution, higher degree of user participation, and an overall agile approach to the development process.

Why are those approaches to service composition and business process design and management relevant at all? Why is it not enough to use workflow management systems? Or is it enough? Well, in this paper, I argue that those traditional approaches increasingly don’t work. The reasoning is as follows: Throughout the last decades we have seen that organizational boundaries increasingly became fuzzy. Novel business alliances, including mergers and acquisitions, are occurring. Such partnerships happen more often and faster than previously. Furthermore, partnerships need to be highly dynamic and flexible, often depending on special cases and on-demand policies. In technical terms we can say that there is increasingly a need for information systems integration, however, the assumptions as we knew them from the area of workflow management systems (e.g., first you model, then you execute; after exceptions occur, remodel your process and enact again) do not hold any longer due to the requirements of highly dynamic, flexible and inter-connected organizations and people including the products and services they offer, provide and produce. The distinction between design (model or built) time and run time is starting to become obsolete. We need to spend more energy on analyzing finer “granularities” of those “times”.

The remainder of the paper is organized as follows. Section 2 discusses our assumptions with regard to current technology trends and summarizes lessons-learned from four areas which are crucial to the topic of this paper, i.e., Infrastructure Evolution, Software Evolution, Process Evolution, and Teamwork Evolution. Section 3 motivates the approaches chosen with an illustrative example. Section 4 discusses the technical approaches we use in our research to solve those presented challenges. Section 5 concludes the paper.

2 Assumptions

Before we propose technical approaches and argue why it makes sense to move towards autonomic processes and services, we outline our assumptions on the relevant technological landscape and context. The devices we use increasingly become smaller, more powerful, cheaper, and always connected to networks. Basically, we move towards a pervasive communications paradigm, where people are enabled to communicate and coordinate their work activities anytime and from anywhere, potentially with many devices. Such as pervasive underlying infrastructure model implies the need for an efficient utilization model for hardware resources (e.g., Grid computing) and software resources (e.g., Service-oriented Computing). The fundamental assumption in this domain is that we increasingly have complex, open and dynamic infrastructures where business processes and services have to operate on. We summarize our assumptions on four dimensions and subsequently discuss our contributions and challenges in them.

(1) **Infrastructure Evolution**. Complex, open and dynamic infrastructures require all of their constituents to operate, to communicate, and to coordinate constantly, in order to keep the overall system in a healthy mode. We can say that this operational principle has some similarity with the human body, the autonomous nervous system, respectively. Therefore, some research communities refer to this research domain as “Autonomic Computing”. However, currently the scientific community working in this domain mainly focuses on lower layers of the software (e.g., operating systems) and hardware stacks (e.g., networking) and intends to add autonomic features to the underlying infrastructure including what is referred to as the self-* properties (e.g., self-healing, self-configuring, self-adapting, self-organizing, self-optimizing, etc.). We should note that not only the underlying infrastructure is supposed to act autonomically; also higher levels of the software stack need to be composed accordingly. To understand what the requirements for such a higher level autonomic composition of processes and services are, we discuss three main lessons learned from the most important parts of autonomic processes and services:

(2) **Software Evolution**. Software requirements cannot be fully gathered upfront or be frozen. Requirements are intrinsically decentralized and a complete control and pre-plan are illusory. When software is changed, it impacts the whole product, process, and service. Software Evolution is intrinsic to software it is not a “post-delivery” nuisance. We basically have two strategies to deal with mastering the complexity of software evolution: a “top-down approach by (a) using *process-driven and model-driven approaches* to master complexity and enterprise-scale change. This means that we build a (process and service composition) model once and use it many times; or a “bottom-up approach” (b) by using end user-driven composition or *service mashups* for small-scale processes and service compositions (i.e., build once and use once).

(3) **Process Evolution**. When we analyze business processes today we see that they typically go across multiple departments, potentially over multiple organizations and countries and run on multiple systems. Unlike databases, where one can query and ask for all customer order info, it is very difficult or impossible to query such “process” related questions. The reason is that business processes are instantiated not on one system only (e.g., a DBMS) but rather leave traces in a plethora of information systems, including workflow systems, databases, mail servers, document management systems, web servers, and mail servers, just to name a few prominent examples. If we require mechanisms to (semi) automatically adjust processes and service compositions to new circumstances – and this is what the underlying assumption here is – we require better abstractions and systems to allow us to do so. It is simply not sufficient to make changes in, e.g., a workflow system since a process touches multiple systems and affects them as well. Making those changes manually does not scale.

(4) **Teamwork Evolution**. Over the past decades teamwork has evolved, both in style and in form. “Classic” teamwork often involved solely intra-departmental work with stable team configurations (i.e., team members did not change frequently) and with long-lived time span (i.e., team members worked together over many years). With the advent of the Internet, and the Web as a communications and collaboration platform in particular, teamwork evolved into what is known as “virtual teamwork”. This essentially means that a more or less stable team usually from different organizations works together for a limited amount of time (e.g., project-based). More recently, we find more team forms, including nomadic teams (i.e., teams on the move) and nimble teams (e.g., a team consisting of specialists to solve a particular problem). Both of the latter team forms have in common that the team configuration may change rapidly and often (e.g., due to network issues in MANETs or due to specialists joining or leaving the team after they accomplished their mission).

Those categories of evolution (Infrastructure, Software, Process, Teamwork) require novel strategies to deal with the design and enactment of supporting infrastructures and information systems. Those novel strategies include self-* capabilities of the underlying infrastructure on the one hand but also autonomic mechanisms on higher levels of abstractions, including the business process levels and service composition levels.

3 Illustrating example

In order to motivate the need for autonomic processes and services consider the following example system: credit management system. Such a system typically provides answers to questions such as: which credit is the right one for me? Credit management is part of a larger system since it depends on issues such as various insurance mechanisms, various repay models, legal and business regulations and many models and regulations more. To summarize: The overall system for managing such credit management features is inherently open, complex, and distributed because interest rates, the status (context) of the credit taker (e.g., illness, insolvency etc.) all have impact on the credit model and rates. The question is how should such an information system be modeled? We argue that we require novel abstractions and mechanisms to solve the problems in such open, complex and distributed scenarios.

4 Technical Approaches for Autonomic Processes and Services

As we have seen, to master complexity in information systems one requires strong links between the parts of the systems (similar to the human autonomous nervous system). Those relationships provide a fundamental framework for the processes and service compositions to be “glued” together in a flexible and adaptive manner.

In our research group, we contribute to the field of autonomic processes and services with the following approaches, methods, and tools we develop: (1) Model-driven compliance framework and approach, (2) Active service registries, (3) Service search and clustering engines, and (4) Context-based and relevance-based service composition and enactment.

4.1 Model-driven compliance framework

In this research [1] we contribute with a view-based and model-driven development (MDD) approach to reduce the development complexity of the overall autonomic systems. The framework consists of modeling elements such as a meta-meta-model, meta-models, and views. As mentioned in the previous section, a view is a representation of a process from the perspective of related concerns. In our framework, a view is specified using an adequate framework's meta-model. Each meta-model is a (semi-)formalized representation of a particular business process concern. Therefore, the meta-model specifies entities and their relationships that can appear in the correspondent view. The meta-models, in turn, are defined on top of the meta-meta-model. The meta-meta-model can be simple or more elaborate like MOF.

4.2 Active Service Registries

In our research in active service registries [2] we address one fundamental shortcoming of today’s SOA implementations, namely, dynamic binding and invocation. We illustrate the set of today’s challenges by utilizing an example based on which those shortcomings are analyzed henceforth. SOAs had foreseen the publish-find/bind cycle (SOA triangle), whereas as today, most SOA implementations use (for practical reasons) only the interaction between service requestor and service provider with service contracts. This, of course, limits the envisaged potential of SOA implementations considerably. In our research project VReSCO we provide a client-side API to allow for dynamic binding and invocation of services to solve many of today’s problem related to dynamic binding and invocation and its relationship to registries. In this paper we discuss those implemented parts of our infrastructure which can be of help when building large-scale SOAs requiring dynamic binding and invocation.

4.3 Service search and clustering engines

In our research on service search engines and clustering [8] we presented a novel distributed Web service search engine based on the Vector Space Model for information retrieval. We have shown that our prototype implementation works even for large WSDL repositories. Unlike other search engines, no template document collection exists to evaluate the final precision/recall rating. To formally evaluate and optimize the search engine’s performance parameters, a test-collection with predefined results has to be established. Furthermore, the vector matrix is currently uncompressed. By erasing zero entries in the matrix and therefore compressing the vector space, we think the performance can be increased significantly. We think that it is very hard to automatically generate working applications out of Web services without human judgment. Creating ontologies may help to a limited degree. For the future, we plan to extend the indexing procedure from purely syntactical data to a semantic level. For this purpose we will utilize a domain-specific ontology to describe the functionality of a service endpoint and integrate the result in a BPEL-process. The major problem here is, to find a fitting indexing method for the ontology itself. Furthermore, by using a domain-specific resource, the application domain is limited equally, which is quite the opposite of what we want to achieve. A possible tradeoff could be achieved by combining syntactical analysis and ontology-supported weight adjustment. It remains to be seen how beneficial the application of ontologies is to leverage the search mechanism to a semantic level.

4.4 Context- and relevance-based service composition and enactment

The inContext EU FP6 research project [3] aims at supporting highly dynamic forms of human collaboration such as Nimble (short-lived collaboration to solve emerging problems), Virtual (spanning different geographical places and involving diverse professionals) and Mobile (collaboration with mobility capabilities) teams. These teams require different mechanisms for coordination, and in many cases also different software services (e.g., document sharing, project management, and instant messaging), and infrastructures (e.g., large-scale and Internet-based mobile devices, and mobile ad-hoc/P2P networks). SOA-based solutions thus offer greater advantages for inContext over other solutions, such as those that are portal-based. For purposes of autonomic services and processes we developed methods to react and to anticipate to changes. This is of paramount importance in autonomic environments. The service adaptation can be based on *context information* (e.g., degradation of QoS values) [4, 5], based on *human activity mining* [6] and on *service interaction mining* [7].

5 Conclusion

In this paper we discussed conceptual challenges as well as technical issues regarding advancements towards autonomic processes and services. We have motivated the need for systems capable of autonomic behavior by looking at the business demands and technological advances, which have changed significantly over the last decades. We outlined four research areas where we summarized our assumptions in more detail: Infrastructure, Software, Processes, and Teamwork. Finally, we presented a summary of our approaches which enable building a coherent framework for autonomic processes and services. We observe that service mashups have an impact in the software evolution domain, which helps to address the dynamics of infrastructures, team forms, and process evolutions, while traditional service composition (e.g., Model-driven development, or MDD for short) will help to address the complexity (e.g., interoperability, multiple platforms, etc). Eventually, in the future service mashup approaches require a “lightweight/on-demand MDD” support to help addressing the dynamics, while still ensures solving the “complexity” issues.

6 References

[1] Tran, H., Zdun, U., Dustdar, S. (2007). View-based and Model-driven Approach for Reducing the Development Complexity in Process-Driven SOA, International Conference on Business Processes and Services Computing, 25-26 September, Leipzig, Germany

[2] Michlmayr, A. Rosenberg, F., Platzer, C., Treiber, M., Dustdar, S. (2007). Towards Recovering the Broken SOA Triangle - A Software Engineering Perspective", In Proceedings of the 2nd International Workshop on Service-oriented Software Engineering (IW-SOSWE'07), Dubrovnik, Croatia, September 2007, ACM Press.

[3] http://www.in-context.eu

[4] Rosenberg, F., Platzer, C., Dustdar, S., (2006). Bootstrapping Performance and Dependability Attributes of Web Services. IEEE International Conference on Web Services (ICWS'06), 18. - 22. September 2006, Chicago, USA.

[5] Rosenberg, F., Platzer, C., Dustdar, S., (2007). QUATSCH – A QoS Evaluation and Monitoring Tool for Web Services. Journal on Web services Research, forthcoming

[6] Dustdar, S., Hoffmann, T. (2007). Interaction pattern detection in process oriented information systems, Data and Knowledge Engineering, Elsevier, 62 (2007), pp. 138–155

[7] Dustdar, S., Gombotz, R. (2007). Discovering Web service workflows using Web services Interaction Mining. International Journal of Business Process Integration and Management (IJBPIM), pp. 256-266.

[8] Platzer, C., Dustdar, S. (2005). A Vector Space Search Engine for Web Services, IEEE European Conference on Web services (ECOWS), 14-16 November 2005, IEEE Computer Society Press.