On the Necessity of an Architecture Framework for On-The-Fly Computing

Bahar Jazayeri
Paderborn University
Paderborn, Germany
bahar.jazayeri@upb.de

Simon Schwichtenberg
Paderborn University
s-lab – Software Quality Lab
Paderborn, Germany
simon.schwichtenberg@upb.de

***Abstract*—In our research, we investigate design processes of future software systems. Our vision is that software services are provided in a world-wide distributed market and are composed automatically in order to fulfil individual user requests on-the-fly. This On-The-Fly Computing (OTF) software provision can be defined and then applied in different domains taking into account constraints of those domains. In practice, such constraints are interrelated and distributed among business models, software applications, and technical infrastructures. Thus, an efficient, holistic, and consistent realization of the OTF paradigm in different domains demands a well-defined architecture that reveals the main building blocks of OTF. In this paper, we discuss the suitability of existing architecture frameworks and derive the requirements for an architecture framework to realize OTF in different domains. The architecture framework considers dependencies among design constraints in final software. This knowledge helps architects to systematically apply the principles of OTF computing in their own domains in practice.**

I. Introduction

In our future vision of software systems[1], distributed world-wide markets provide software services to individual user requests automatically and on-the-fly. We call such an enhanced provision of software services On-The-Fly (OTF) computing. The OTF paradigm is to be enabled by automatic discovery and composition of single services that are available in global markets. To achieve the desired service delivery, several stakeholders contribute to OTF computing such that each plays a certain role, e.g., service provider, service composer, quality checker, etc. However, each of these stakeholders may have different configurations and realization in different domains. Therefore, OTF systems can be differently customized in certain domains taking into account the constraints of each domain. In practice, such domain constraints are interrelated and distributed among business models, software applications, and technical infrastructures. For instance, service providers may need to be known and certified in in-house domains, while in public domains, they may be allowed to provide services anonymously.

These customizations give rise to realization of a wide range of complex and diverse application of OTF computing in different domains. Examples of software markets in different domains are mobile App stores like Google Play[2], markets of Internet-of-Things services like IFTTT[3], and market of web services like Mashape[4]. Although these markets provide different types software services, the application of OTF computing can highly improve service provision in these markets, e.g., by adding automatic and on-the-fly service composition [1]. Thus, an efficient, holistic, and consistent application of OTF computing requires a well-defined generalization at the architecture level. Existing architectural frameworks distinguish between different types of architectures like business, software application, and infrastructure architectures. However, in practice the architectural constraints highly influence each other. For example, the constraints related to infrastructure and execution resources may be customized based on the business model of the market. Furthermore, when a change in requirements happens, the rationales behind architectural and domain constraints become hard to detect for development teams, because the cross-relations between architectural constraints is not foreseen in the architecture.

In this paper, we discuss the suitability of existing architecture frameworks for realization of OTF computing in practice. Furthermore, we derive the requirements for an architecture framework to realize OTF concepts in different domains. In the following, related work is considered in Section II. Section III introduces the challenges, which we need to deal with in defining an architecture framework for OTF systems. In Section IV, we derive our first ideas for an appropriate architecture framework for a solid realization of OTF computing.

II. Related work

Although there is a large body of work to define the architecture of complex and heterogeneous systems like OTF systems, none of them considers both the interrelation between business and IT aspect and the main building blocks of OTF computing.

Architecture frameworks encompass the family of architectures that a software system can have. They describe the business-, software-, and infrastructure architectural elements and their relationships. Examples of architecture frameworks are Zachman [2], TOGAF [3], and IAF [4].

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing"(CRC 901).

[1]For more information refer to http://sfb901.uni-paderborn.de

[2]play.google.com
[3]ifttt.com
[4]www.mashape.com

Reference architectures, in comparison with architecture framework, describe a concrete architecture that a system with concrete requirements in a specific application domain can be derived from. AUTOSAR for automotive applications [5] and SeAAS as a reference architecture for security services are examples of reference architectures [6].

Architecture description languages, e.g., Interface Definition Language (IDL), Web Service Description Language (WSDL), and UML-based languages are used to describe architecture frameworks or reference architectures.

III. Challenges of defining an architecture framework for OTF Computing

Our study of the problem identifies three main challenges corresponding to the definition of an architecture framework for realization of OTF computing:

1) Main architectural building blocks and their relationships that are needed to enable the core part of OTF computing, i.e., automatic discovery and composition of software services.
2) The tight relations between different types of architectures needs to be defined properly. This consideration need to include the whole life cycle of markets, including design, development, and evolution. Specially, the architecture framework needs to constantly deliver business values.
3) Cross-cutting concerns, distributed across architectures, need to be supported by the architecture framework.

IV. Preliminary Results and Contributions

In this section, we propose the main structure of an appropriate architectural solution for the realization of OTF computing. As mentioned in Section I, several role players support OTF service provision, e.g., service providers, brokers, and OTF compute centers. OTF Architecture Framework consists of several components as following: it includes three architectures: business, application, and infrastructure architectures. *OTF Business Architecture* consists of business models of OTF markets. *OTF Application Architecture* includes applications and software systems. The application architecture includes the core part of sophisticated software enabling OTF computing, i.e., matcher, composition engine, configurator, etc. Moreover, *OTF Infrastructure Architecture* holds infrastructure, e.g., networking resources and hardware.

Another component of OTF Architecture Framework manages *Cross-Cutting Concerns* of OTF systems using aspect-oriented programming [7]. Cross-Cutting Concerns are aspects distributed across several architectures. A typical example of Cross-Cutting Concerns is security, which can be security requirements in business model, software applications, or at infrastructure of a market.

Moreover, the OTF Architecture Framework includes *Architecture Dependency Management*, which is responsible for handling dependencies between the sub-architectures. To perform this task, it keeps all mandatory and also variable features of the architecture framework [8]. Architecture Dependency Management improves the awareness regarding the dependencies between business, software, and infrastructure architectures. This decreases the chance that the rationales and knowledge behind the architectural constraints become vaporized during system evolution. In addition, the dependency management improves the architect's decision-making in large spaces of highly dependent design decisions. Using this knowledge, the architects identify the trade-off between quality attributes of the final design in an efficient way.

Finally, *OTF Architecture Governance* provides analytic methods to check the compliance of a system, which applies OTF computing, with the guidelines and principles defined in the architecture framework. The compliance checking proves whether the core OTF building blocks and their interrelations are correctly realized in the system. Furthermore, the structure of system needs to conform to business strategies. This, for instance, includes whether services are accessible outside of an organization or whether the intellectual property of service providers need to be protected by an OTF system. Afterwards, the degree of maturity for a market is calculated using a quality model based on the results of the compliance-checking. In addition, possible improvements to the market structure are provided based on this analysis.

V. Summary

In this position paper, we introduce the necessity of defining an architecture framework for the application of On-The-Fly (OTF) computing in different domains. The identification of architecture framework helps to realize the enhancements like automatic service discovery and composition in practice. The architecture framework includes three architectures, i.e., business, application, and infrastructure architectures. This considers the tight and continuous interrelations between architectural design decisions and the management of cross-cutting concerns among these architectures. In the future, main quality attributes and differentiators of different domains and domain-specific challenges of applying the principles of OTF architecture framework can be further identified.

References

[1] Bahar Jazayeri and Simon Schwichtenberg. On-The-Fly Computing Meets IoT Markets Towards a Reference Architecture. In *International Conference on Software Architecture Workshops*. IEEE, 2017. (to appear).

[2] John Zachman et al. A framework for information systems architecture. *IBM systems journal*, 26(3):276–292, 1987.

[3] Van Haren. Togaf version 9.1. 2011.

[4] Jack Van't Wout, Maarten Waage, Herman Hartman, Max Stahlecker, and Aaldert Hofman. *The integrated architecture framework explained: why, what, how*. Springer, 2010.

[5] AUTOSAR. http://www.autosar.org/specifications. Last Access: April 2017.

[6] Michael Hafner, Mukhtiar Memon, and Ruth Breu. Seaas-a reference architecture for security services in soa. *J. UCS*, 15(15):2916–2936, 2009.

[7] Elisa Yumi Nakagawa, Rafael Messias Martins, K Felizardo, and Jose Carlos Maldonado. Towards a process to design aspect-oriented reference architectures. In *Proceedings of the XXXV Latin American Informatics Conference (CLEI 2009)*, pages 1–10, 2009.

[8] Paul Clements, Linda Northrop, et al. A framework for software product line practice. *SEI interactive*, 2(3), 1999.