Mora, M., O'Connor, R. V., Tsui, F., and Marx Gómez, J. (2018) Design methods for software architectures in the service-oriented computing and cloud paradigms. Softw. Pract. Exper., 48: 263–267. doi: 10.1002/spe.2547.

Design methods for software architectures in the service-oriented computing and cloud paradigms

M. Mora[1] , R. V. O'Connor[2], F. Tsui[3] and J. Marx Gómez[4]

[1] Department of Computing Sciences, Autonomous University of Aguascalientes, Ave. Universidad 940, Aguascalientes 20131, AGS, Mexico

[2] School of Computing, Dublin City University, Glasnevin, Dublin, Ireland

[3] Department of Software Engineering and Game Design, Kennesaw State University, 1000 Chastain Road, Kennesaw, Georgia 30144-5588, USA
[4] Department of Informatics, Carl von Ossietzky University Oldenburg, Ammerländer Heerstr. 114-118, Oldenburg 26129, Germany

The design of the system architecture has been a critical and relevant activity in the systems engineering discipline from several decades for the design of physical man-made systems [1]. However, a reduced set of design methods and complementary decision-making techniques have been consistently used with successful results [2–4].

In contrast, in the software engineering discipline, there is a body of academic research on the foundations, methods and complementary techniques with a scarce accumulative knowledge. Every design stream on software architecture elaborates its own vocabulary and methods and usually ignores other design streams [5]. Additionally, there are more proprietary than open access design methods, and a reduced set of well-accepted methods, techniques and practices are still missing [6]. Thus, software architecture design methods are inconsistently or little used in the practice during the application of software development methodologies except by the large system and software development companies [7, 8]. As a pioneer [9] in the software architecture domain indicates (idem, p. 29)

> *However, despite this progress, as engineering disciplines go, the field of software architecture remains relatively immature. While the foundations of an engineering basis for software architecture are now clear, there remain numerous challenges and unknowns.*

In particular, software systems are widely demanded at present in two co-existing paradigms: Software-oriented Computing (SOC) [10, 11] and the Cloud Computing paradigms [12, 13]. SOC paradigm aims the development of medium-sized and large-distributed inter-organizational systems by assembling it (ideally) as a system of service providers and consumers, known as a Service-oriented Computing Application (SOCA) [14]. Thus, the design of SOCAs inherently includes the challenges of high-quality software architecture design, given its natural composition by services provided by some entities and consumed by others (independently of the self-services in the same SOCA). In turn, the Cloud Computing paradigm aims to provide a cost-effective, scalable and dynamic provision of IT resources (processing, storage, networking) for elastic demands. This convergence of SOC and this emergent paradigm introduces new software architecture design challenges and trade-off problems for achieving functional and non-functional requirements of Software as Service (SaaS) as a whole entity assembled by other composed and atomic services in the cloud [14, 15]. Additionally, main Systems Engineering architectural successful design patterns (hierarchical and functional-driven ones) differ from the main ones from the Software Engineer domain (layered and data-driven ones) [16].

Hence, we consider that 1) the SOC and Cloud Computing applications strongly depend of their architectural designs; 2) practitioners demand well-structured and practically applicable software architecture design methods; 3) there is a scarcity of well-accepted software architecture design methods; and 4) there is a wide gap between theoretical recommendations and their daily utilization in the practice for software engineers working mainly in small and medium-sized companies.

Thus, we considered that this call for papers is relevant and useful for collecting high-quality conceptual proposals or already applied software architecture design methods for SOC and Cloud Computing paradigms.

This special issue aimed lately to improve our understanding and encourage the utilization of software architecture design methods by being perceived as useful, ease of use, ease of learning, compatible, and highly valued by practitioners. For it, we called for papers on the following topics:

- Case studies of successful applied software architecture design methods for SOC and Cloud Computing paradigms
- Case studies off ailed utilizations of software architecture design for SOC and CloudComputing paradigms
- Usability studies on specific software architecture design methods for SOC and Cloud Computing paradigms
- Successful examples of inclusion of software architecture design methods into agile software development methodologies for SOC and Cloud Computing paradigms from practitioner's perspectives
- Software architecture design methods for SOC and Cloud Computing paradigms based on ISO/IEC standards (e.g. ISO 42010, ISO 12207, ISO 15288, among others)
- Comparative studies of most recognized software architecture design methods for SOC and Cloud Computing paradigms
- Cases on the development of prototype tools for supporting the software architecture design tasks for SOC and Cloud Computing paradigms.

This special issue requested original applied research papers that challenge or confirm the current status of software architecture design methods and complementary techniques through the report of usable well-structured software architecture design methods in the SOC and Cloud Computing paradigms. Papers based on original ideas were expected. Papers elaborated as extended versions of previous published research had to provide at least 50% of new material. For this case, authors had to submit a cover letter/document with 1) a comparative table of findings, contributions and limitations from extended and initial paper; and 2) a brief justification on the extended sections in the new paper. Each submitted paper was carefully reviewed by two international reviewers at least. The editorial review committee was composed by academic and industry-based people involved in the special issue themes. Our paper selection and evaluation criteria were

- Evidences of advances on the current status of scholastic software architecture design methods
- Practical relevance to use and apply the method or contributions of the paper
- Scope and criticality of the application domain of the method or contributions of the paper
- Correctness of the applied research method (problem definition, work hypothesis, materials and methods, application of materials and methods, discussion of results, contributions and limitations, and conclusions)
- Clarity and readability of the paper

Finally, the accepted papers had to demonstrate clearly how their contributions advance the current status of scholastic software architecture design methods.
We had a positive response from international research communities involved in Software Architecture Design Methods for SOC/SOA and Cloud Computing. After an approximate 2-year period of several rounds of reviews and corrections on conditioned papers, we finally accepted the following 5 high-quality papers:

- DOI: 10.1002/spe.2517. The Architectural Template Method: Templating Architectural Knowledge to Efficiently Conduct Quality-of-Service Analyses.
- DOI: 10.1002/spe.2543. An Iteration-based Interactive Analysis Method to Design Dynamic Service-oriented Systems.
- DOI: 10.1002/spe.2457. Dynamic Reconfiguration of Cloud Application Architectures.
- DOI:10.1002/spe.2496.ArchitecturalPartitioningandDeploymentModelingonHybridClouds.
- DOI: 10.1002/spe.2472. JDAS: A Software Development Framework for Multidatabases.

The first paper is entitled "The Architectural Template Method: Templating Architectural Knowledge to Efficiently Conduct Quality-of-Service Analyses" and was co-authored by Sebastian Lehrig at Paderborn University, Germany, and Marcus Hilbrich and Steffen Becker both at Chemnitz University of Technology, Germany. In this paper, the authors provide a new method to support the reuse of architectural templates which contains relevant knowledge required for making architectural design decisions on SOA systems. These templates can be instanced by designers with specific elements for a particular design case. Authors provide practical evidences on the benefits of applying this new method through a realistic case study: a book e-store open- system migration from a web to a cloud platform. In this case study, the authors analyze QoS properties of performance, scalability, elasticity and cost-efficiency, on several design decisions, which were systematically conducted with the guidance of this new method. Finally, authors report that both expert and novice software architects can get benefits by tuning their own expertise through the utilization/adaption of the architectural templates (experts), and by learning and keeping a standard design (novices).

The second paper is entitled "An Iteration-based Interactive Analysis Method to Design Dynamic Service-oriented Systems" and was co-authored by Wuping Xiet, Jinyun Xue and Dongming Jiang at Jiangxi Normal University and Wuhan University, China, and Lan

Song at Wuhan University, China. The authors indicate that the total and correct automatization of service composition for a service-oriented system is still an idealized goal due to several practical restrictions. Thus, they propose a heuristic method that combine manual with automated procedures for such a goal. Their method fosters also the reutilization of valuable architectural knowledge and provides a systematic guidance for a satisfactory design. For it, the authors propose the concept of the uncertain candidate service set (UCSS), which is a linkage between the creative (non-automatable tasks) and non-creative (automatable) tasks. The authors illustrate the utilization of the new method by using a realistic case study: an SOA travel agency system. Benefits on reutilization and a systematic design are finally reported.

The third paper is entitled "Dynamic Reconfiguration of Cloud Application Architectures" and was co-authored by Miguel Zúñiga-Prieto at the Universitat Politècnica de València, Spain, and Universidad de Cuenca, Ecuador; Javier González-Huerta at Blekinge Institute of Technology, Sweden; and Emilio Insfran and Silvia Abrahão both at the Universitat Politècnica de València, Spain. The authors address the problem of dynamically reconfiguring the architecture of current cloud-based applications by the need of accomplishing new service level agreements (SLAs). This problem extends the problem of scaling assigned resources in the cloud with the reconfiguration of the architecture of the system, and their respective analysis of impact on such new expected SLA issues. For evaluating this method, the authors conducted a quasi-experiment with 20 CSc students (15 at PhD level, 2 at MSc level and 3 at BSc level) where the subjects applied the method to a realistic case of study: a cloud-based reservation system. The quantitative evaluation shown satisfactory results on the metrics of usefulness, ease of use and intention of use.

The fourth paper is entitled "Architectural Partitioning and Deployment Modeling on Hybrid Clouds" and was co-authored by Sreekrishnan Venkateswaran at IBM Corporation, India, and Santonu Sarkar at BITS Pilani K.K.Birla Goa Campus, India. The authors report the benefits of deploying hybrid clouds (private and public ones) from control managerial, service performance and economic perspectives, but they also introduce challenges to meet such demands. Thus, they propose a heuristic model for achieving it derived from the empirical analysis of about 60 real hybrid cloud configurations. Their model uses the workload as a key input for predicting required deployment efforts. The authors validated the model with 12 real cases from retail, finance, healthcare and airline sectors, where the authors had participated in the design of hybrid cloud solutions. The predicted deployment efforts with the heuristic model were reasonably near to real values. This model, thus, contributes to the design of run-time architectures in hybrid cloud environments.

Finally, the fifth paper is entitled "JDAS: A Software Development Framework for Multidatabases" and was co-authored by Guoqi Xie, Yuekun Chen, Yan Liu, Chunnian Fan, Renfa Li at Hunan University, Key Laboratory for Embedded and Network Computing of Hunan Province, and Nanjing University of Information Science and Technology, China; and Keqin Li at the State University of New York, USA. The authors indicate that the trend for modern SOA or Cloud-based systems is not to rely in a single database system but a distributed multidatabase system environment. Thus, specific architectural design decisions on the database systems to be used must be considered. The authors report a Java-based development framework called Java data access service

(JDAS) for multidatabases, which reduces design complexity and improves design efficiency. The authors reported two experiments for comparing the Spring/JDBC, Spring/MyBatis, Spring/Hibernate and JDAS frameworks regarding lines of code and size in MBs, and a case of study on an e-government system. In the three evaluations, the application of the JDAS produced satisfactory results and improved the specific evaluated metrics regarding the other three development frameworks.

Hence, we consider that this special issue contributes clearly with the planned general aim established as "to improve our understanding and encourage the utilization of software architecture design methods by being perceived as useful, ease of use, ease of learn, compatible, and highly valued by practitioners".

We finally thank authors and anonymous reviewers for your excellent academic work, as well as Professors Rajkumar Buyya at the University of Melbourne, Australia, and R. Nigel Horspool at the University of Victoria, Canada, who are the EiCs of Software Practice & Experience (SPE) journal for all academic and guidance provided during this guest editorial process. We also thank Jack Paterson and all SPE journal publishing staff for your assistance on several publishing procedures.

REFERENCES

1. Crawley E, de Weck O, Eppinger S, Magee C, Moses J, Seering W, Schindall J, Wallace D, Whitney S. The Influence of Architecture in Engineering Systems, Engineering Systems Monograph, ESD Architecture Committee (2004), 2004; pp. 1–29, EDS MIT.
2. Pyster A, Olwell D, Hutchison N, Enck S, Anthony J, Henry D, Squires A. Guide to the Systems Engineering Body of Knowledge (SEBoK) version 1.0. Hoboken, NJ: The Trustees of the Stevens Institute of Technology, USA 2012.
3. Rechtin E. Systems Architecting: Creating and Building Complex Systems. Prentice-Hall: USA, 1991.
4. Sage AP, Lynch CL. Systems integration and architecting: an overview of principles, practices, and perspectives. Systems Engineering 1998; 1(3):176–227.
5. Reyes-Delgado PY, Mora M, Duran-Limon HA, Rodríguez-Martínez LC, O'Connor RV, Mendoza-Gonzalez R. The strengths and weaknesses of software architecture design in the RUP, MSF, MBASE and RUP-SOA methodologies: a conceptual review. Computer Standards & Interfaces 2016; 47:24–41.
6. Hofmeister C, Kruchten P, Nord R L, Obbink H, Ran A, America P. A general mode of software architecture design derived from five industrial approaches. Journal of Systems and Software 2007; 80(1):106–126.
7. Lange C, Chaudron M, Muskens J. In practice: UML software architecture and design description. IEEE Software 2006; 23(2):40–46.
8. Weinreich R, Buchgeher G. Towards supporting the software architecture life cycle. Journal of Systems andSoftware 2012; 85(3):546–561.
9. Garlan D. Software architecture: a travelogue. In: Proceedings of the on Future of Software Engineering FOSE'14 (pp. 29–39), May-Jun, Hyderabad, India 2014.
10. Bouguettaya A, Singh M, Huhns M, Sheng QZ, Dong H, Yu Q, Neiat AG, Mistry S, Benatallah B, Medjahed B, Ouzzani M. A service computing manifesto: the next 10 years. Communications of the ACM 2017; 60(4):64–72.
11. Papazoglou MP, Traverso P, Dustdar S, Leymann F. Service-oriented computing: state of the art and research challenges. Computer 2007; 40(11):64–71.

12. Buyya R, Yeo CS, Venugopal, S. Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities. In High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on 2008; (pp. 5–13), September, Dalian, China.
13. Zhang Q, Cheng L, Boutaba R. Cloud computing: state-of-the-art and research challenges. Journal of internet services and applications 2010; 1(1):7–18.
14. Gu Q, Lago P. Guiding the selection of service-oriented software engineering methodologies. Service Oriented Computing and Applications 2011; 5(4):203–223.
15. Sommerville I. Teaching cloud computing: a software engineering perspective. Journal of Systems and Software 2013; 86(9):2330–2332.
16. Maier MW. System and software architecture reconciliation. Systems Engineering 2006; 9(2):146–159.