Business Process Management and Service Oriented Architecture

By Eva Rio

*This content is part of a research paper I wrote for the Service Research Seminar course at Aalto University

***Abstract* — In computer science and engineering, the service orientation paradigm does not contemplate software as a product anymore, but as a service that is divided into smaller pieces that solve different concerns, as in a distributed system. Service orientation pursues the automation of business logic, which can bring several benefits to companies, for instance more agility and scalability, lower costs and interoperability. Thus, the topic is of great interest not only for IT and software companies, but also for academia, who starts showing curiosity now that the scientific community begins to accept more practical research methods (such as design-based science, better suited for the analysis of this topic) as valid scientific approaches for new knowledge creation. Service Oriented Architecture and everything that it entails, is still an uncharted territory full of new possibilities for both scientific and industrial research.**

I. INTRODUCTION

THIS paper reviews basic concepts of service orientation in information systems and computer science and engineering. First, the paper introduces the design-based researching methodology and its characteristics. Next, it gives an overview of the Service Oriented Architecture (SOA) paradigm, its principles, and its architectural foundations. Section IV compares the advantages and disadvantages of the two most well-known approaches to SOA, SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). Section V presents the Business Process Management (BPM) approach and its relation with SOA. The last section recapitulates the most relevant points of all the sections of the paper.

II. DESIGN BASED SCIENCE

Design-based science (DBS) *is an inquiry-based science pedagogy in which new scientific knowledge and problem- solving skills are constructed in the context of designing artifacts* [1]. DBS has different research objectives compared to another approaches: it is more concerned with utility than theory. DBS does not focus on understanding the reality, but on designing and creating artifacts (namely models, methods, constructs and implementations) that serve human purposes [2]. Essentially, DBS consists of constructing new and innovative artifacts to achieve certain goals to solve real-world problems [1, 2]. The two main processes of DBS are building and evaluating [2]. Building is the process of constructing the artifact to prove it works in order to demonstrate feasibility, and evaluating is the process of measuring how well the artifact works in order to demonstrate performance. DBS is especially suitable for conducting research in applied disciplines such as Information Systems [3, 4]. The seven guidelines presented in [4] are a set of recommendations for conducting design- science research in the field of Information Systems.

Elaborating on the findings from [3] and [4], [5] illustrates the three cycles of design-based science as a research approach: relevance, design and rigor (fig. 1).

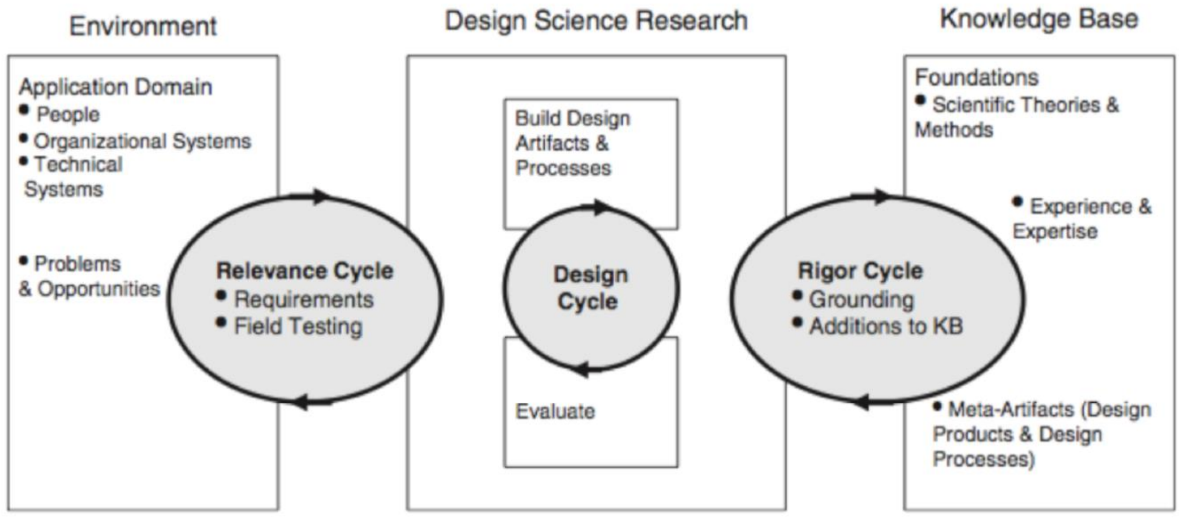


Fig. 1. Design science research cycles [5]

The relevance cycle is the basis for the actual design research process. It provides an overview of the opportunities and problems in the environment (i.e. the requirements) and field testing is performed at this stage to determine the adequacy of the requirements and the need for iterations on the cycle. The design cycle is the core activity of the research process. Based on the requirements from the relevance cycle, several artifacts are built and evaluated in order to find the most suitable alternative. This cycle requires a lot of iteration until the most satisfactory design is achieved. The rigor cycle is performed to ensure that the research generated by the design-based method meets scientific standards and contributes to the knowledge base of science. Justifying the validity of design-based research is not an easy task [4, 5] since it might lack a strong grounding [6] theory (i.e. the research might not be based on any existing theory nor it establishes the basis for a new one) and thus face the rejection from part of the scientific community. Design science also presents an added difficulty: the design of the artifact and its performance depends on the environment in which it operates [2]. Therefore, special attention needs to be given to understanding the environment to avoid incorrect designs or even unsafe artifacts.

III. SERVICE ORIENTED ARCHITECTURE

Service Oriented Architecture (SOA) is a type of software architecture that utilizes services for developing applications in a fast, inexpensive and convenient manner [7, 8]. Taking advantage of certain characteristics of services (especially their uniformity, ubiquity and platform-independency), SOA aims to provide interoperability between languages and platforms to create a network of loosely coupled services in which applications can be developed *on the fly* and *reused everywhere by anyone* [7, 8]. In order to understand how SOA works, it is important to describe the agents that integrate it and their roles. Essentially, there is a service provider which creates and publishes a service, and a service client which has a need and thus searches for a service and once founded, requests for it and uses it. Service aggregators and service operators are two other possible roles: whereas the first one is responsible for combining multiple services into a single offer, the second one has to perform operation management functions of the service [9]. The main characteristics of SOA published in [10] establish a set of design principles for implementing SOA in a system or industry level. Due to the limitations of length in this paper, the eight principles are briefly explained below:

1) Standardized service contract: every service has to abide by an agreement between the service provider and the service client in which the purpose and capabilities of the service are stated.

2) Service loose coupling: services must not depend heavily on each other.

3) Service abstraction: the logic of the service must be hidden.

4) Service reusability: services are designed to work regardless of the context (i.e. for different purposes) without need for rewriting them.

5) Service autonomy: services have a significant degree of control over the execution environment and should not rely on resources that they cannot control.

6) Service statelessness: state data is only accessed when required, thus enhancing scalability.

7) Service discoverability: services need to be easily searchable and identified.

8) Service composability: services can be combined to create a new solution.

Based on these eight characteristics, the architectural structure of service oriented architecture separates the logical functionality of the services into three main layers ([7, 8, 9], fig. 2): *foundations or basic services, composite services, and managed services*.

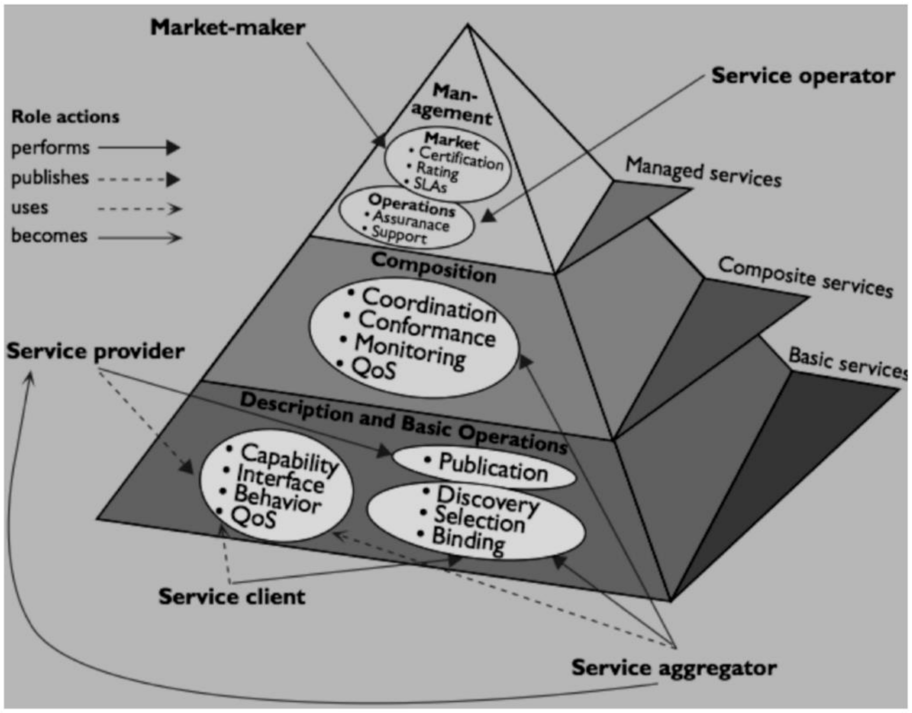


Fig. 2. The SOA pyramid and its layers [7,9]

Every layer defines a different set of functions, roles and relationships, and depends on the previous one to perform its tasks.

1) Foundations or basic services: the first layer of SOA represents the runtime infrastructure of the service, where basic functions (e.g. description, publishing, finding and binding) take place. In this layer, different components and systems are connected and the channel access to service is set.

2) Composite services: the second layer enables the aggregation of multiple services into a single composite service that can be presented as a new solution to the client or used as a basis for other compositions.

3) Managed services: the main purpose of the third layer is to monitor and ensure the performance of the system and high-level aspects such as scalability, availability, or service life-cycle management. This layer also provides support for open service marketplaces, thus enabling the possibility for doing business electronically.

These three layers together with the eight principles constitute the basis for designing and creating services under the SOA approach. At this point, companies can opt for different service realization strategies [7]: in-house, purchasing or leasing, outsourcing, or using wrappers or adapters based on legacy code.

As suggested, service oriented architecture has several advantages (particularly reusability, less programming effort, platform independency and scalability), but it also presents few challenges

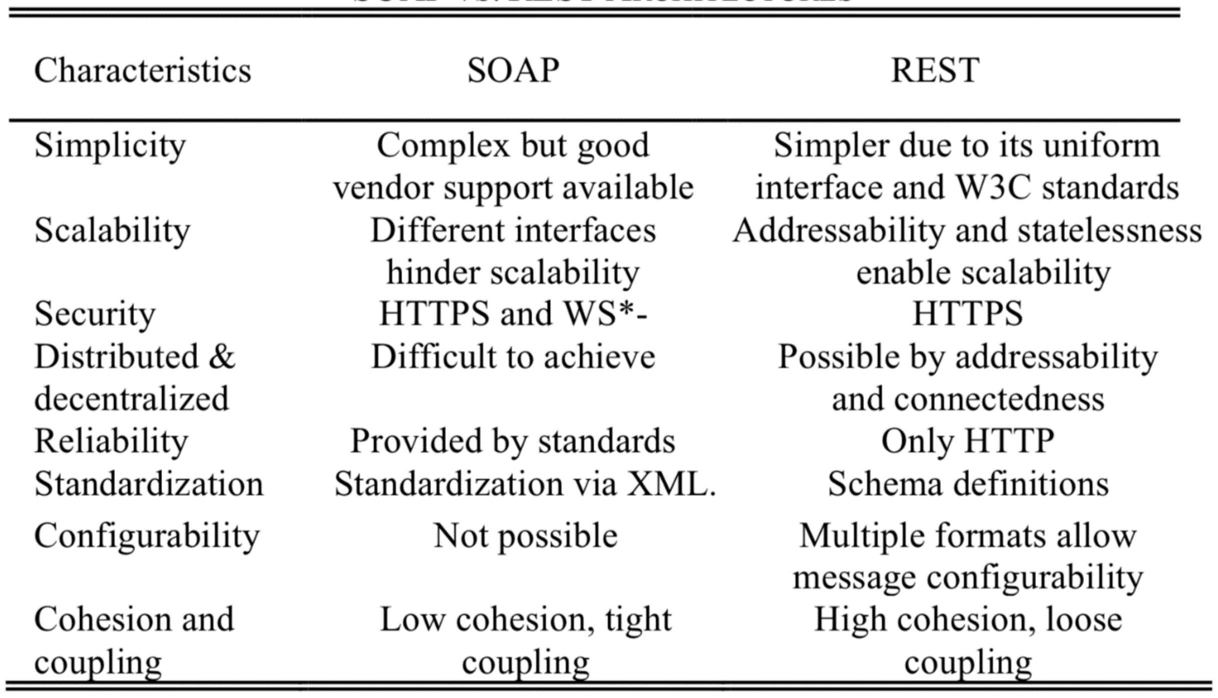
[8], especially in terms of security, the difficulty of aligning SOA practices with business goals and the heterogeneity between current applications.

IV. SOAP AND REST ARCHITECTURES

SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) are the two most popular architectural styles of SOA. Although both of them are used for interfacing with web services (i.e. to enable machine-to- machine communication), their differences are notorious. SOAP is a processing protocol that uses XML for exchanging information through a network [11]. On the other hand, REST is not a protocol, but an architectural style that focuses on the role, constrains and interactions between components to build distributed hypermedia systems [12]. In practice, the main difference between the two of them is how applications or services interact between them: in SOAP, applications interact but they are outside the web (they communicate using WS-* specifications), whereas in REST applications become part of the web (resources, data and services are identified by URIs) [11]. In other words, SOAP sees the web as a transport, and REST as a huge information database. Choosing between one style or another depends on organizational and strategic factors, for instance, the end goal pursued by the company with the SOA implementation, the way internal business processes are managed or the IT resources available.

The table below is based on [13] and summarizes the main characteristics of both architectures:

TABLE I
SOAP VS. REST ARCHITECTURES



Characteristics	SOAP	REST
Simplicity	Complex but good vendor support available	Simpler due to its uniform interface and W3C standards
Scalability	Different interfaces hinder scalability	Addressability and statelessness enable scalability
Security	HTTPS and WS*-	HTTPS
Distributed & decentralized	Difficult to achieve	Possible by addressability and connectedness
Reliability	Provided by standards	Only HTTP
Standardization	Standardization via XML.	Schema definitions
Configurability	Not possible	Multiple formats allow message configurability
Cohesion and coupling	Low cohesion, tight coupling	High cohesion, loose coupling

Both approaches have advantages and disadvantages. Based on the characteristics mentioned above, several authors (e.g. [11], [13], [14]) recommend SOAP for enterprise systems integration (it has good support, requires less development effort and its considered more reliable and secure) and REST for web services solutions (it is more flexible and scalable and information can be cached because it is stateless).

V. BUSINESS PROCESS MANAGEMENT AND SOA

Business Process Management (BPM) is considered a new approach in business management that enables business agility and better operational performance [15]. According to several authors [e.g. 15, 16, 17] its focus on processes makes it a very powerful tool when combined with SOA. As mentioned in section III, SOA allows for the design of software that can be easy reutilized to create new services or for different purposes. With BPM, processes can be analyzed to discover which components (applications, systems) of the company should be re-engineered into SOA to increase efficiency and lower costs [15, 16]. The combination of BPM and SOA helps align business processes and IT resources [15], a strategic topic [18] that has been discussed for years usually under the discipline of IT governance. The purpose of joining BPM and SOA is to assist the needs of dynamic business processes [15], by adapting rapidly to changes and enabling scalability. The two architectural approaches to SOA, SOAP and REST can be used to support BPM strategies. It was discussed in section IV that SOAP is been largely used in an enterprise level and REST is more oriented to web services. Nevertheless, the increasing interest in REST processes has recently led researchers to consider a combination of BPM and RESTful services [19]. More research should be done in this area to discover which of the two architectural systems will better support BPM to transform this union into a source of competitive advantage. As an example, the design-based research method introduced in section I could be used to investigate this topic. Different artifacts (e.g. models or methods) can be designed and implemented to evaluate the appropriateness level of the SOAP and REST architectures. Studying BPM using a design-based methodology has already proved satisfactory in terms of generating new knowledge [20]. The benefits provided by BPM and SOA together are worthwhile exploring.

VI. CONCLUSIONS

Service oriented architecture proposes a convenient way for developing applications that are platform-independent, interoperable and reusable. SOAP and REST are the two main architectural styles that support the service oriented paradigm. SOAP uses the WS*- standards and contemplates the web as a transport, whereas REST is based on the W3C and sees the web as a database. These two types of architectures cannot only be used for creating web services, but also on an industry level. Business Process Management (BPM) is one example of it. Combining SOA and BPM allows companies to gain more agility and flexibility while automating dynamic processes. Little research has been done in this area, but the increasing popularity and acceptance of research approaches like design-based science can contribute to broaden the knowledge in this area.

REFERENCES

[1] D. Fortus, J. Krajcik, R. C. Dershimer, R. W. Marx, R. Mamlok- Naaman, “Design-based science and real-world problem-solving”, *International Journal of Science Education,* vol. 27, no 7, June 2005, pp. 855–879.

[2] S. T. March, G. F. Smith, “Design and natural science research on information technology”, *Decision Support Systems*, vol. 15, no 4, Dec. 1995, pp. 251–266.

[3] J. Iivari, “A paradigmatic analysis of information systems as a design science”, *Scandinavian Journal of Information Systems,* vol. 19 no 2, 2007, pp. 39–64.

[4] A. R. Hevner, S. T. March, J. Park, S. Ram, “Design Science in Information Systems Research”, *MIS Quarterly*, vol. 28, no 1, March 2004, pp. 75–105.

[5] A. R. Hevner, “A three cycle view of design science research”, *Scandinavian Journal of Information Systems,* vol. 19 no 2, 2007 pp. 87–92.

[6] B. G. Glaser and A. L. Strauss, *The discovery of grounded theory: strategies for qualitative research,* New Jersey: Aldine Transaction, 1967, ch. 1.

[7] M. P . Papazoglou, “Service-Oriented Computing: Concepts, Characteristics and Directions”, *4th International Conference on Web Information Systems Engineering* (WISE’03), Rome, Italy, 2003.

[8] M. P Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, “Service- oriented computing: a research roadmap”. *International. Journal of Cooperative Information Systems*, vol. 17, no 2, 2008, pp. 233–255.

[9] M. P. Papazoglou and D. Georgakopoulus, “Service Oriented Computing”, *Communications of the ACM,* vol. 46, no, 10, Oct. 2003, pp. 25–28.

[10] T. Erl, *SOA: Principles of Service Design,*Indiana,USA:Prentice Hall, 2007, ch. 4.

[11] C. Pautasso, O. Zimmermann, and F. Leymann, “RESTful Web services vs. ”big” Web services: making the right architectural decision”, W*WW ’08: Proceeding of the 17th international conference on World Wide Web,* Beijing, 2008.

[12] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures”, *Doctoral dissertation,* University of California, Irvine, 2000, ch. 5.

[13] J.Becker, M.Matznerand, andO. Muller,“Comparing architectural styles for service-oriented Architectures: a REST vs. SOAP case study”, in *Information Systems Development: Towards a Service Provision Society,* New York: Springer, 2009, ch. 22.

[14] P. A. Castillo, J. L. Bernier, M. G. Arenas, J. J. Merelo, P. García-Sánchez, "SOAP vs REST: comparing a master-slave GA implementation", arXiv:1105.4978v1, May 25, 2011.

[15] P. Malinverno, J. B. Hill, "SOA and BPM are better together", *Gartner Research*, Feb. 9, 2007.

[16] G. K. Behara, "BPM and SOA: a strategic alliance", *BP Trends*, May 2006, www.bptrends.com/publicationfiles/05-06-WP-BPM-SOA-Behara.pdf

[17] F. Kamoun, "A Roadmap towards the Convergence of Business Process Management and Service Oriented Architecture", *Magazine Ubiquity*, New York: ACM, April 2007.

[18] J. C. Henderson and N. Venkatraman, "Strategic alignment: leveraging information technology for transforming organizations". *IBM Systems Journal*, IBM Corporation, vol. 38 no 2-3, pp. 472-484, 1993.

[19] C. Pautasso, "Business Process Management with REST", *SlideShare presentation*, Faculty of Informatics, USI Lugano, 2010, https://www.slideshare.net/cesare.pautasso/bpm-with-rest

[20] M. Heravizadeh, "Quality-aware Business Process Management", PhD thesis, *Queensland University of Technology*, 2009.