Characterization of the Application of Service-Oriented Design Principles in Practice: A Systematic Literature Review

Michel S. Soares[1*], Joyce M. S. França[2]

[1] Federal University of Sergipe, São Cristóvão, Brazil.
[2] Federal University of Uberlândia, Uberlândia, Brazil.

* Corresponding author. Tel.: 55 79 991272235; email: mics.soares@gmail.com
Manuscript submitted November 5, 2015; accepted February 20, 2016.
doi: 10.17706/jsw.11.4.403-417

Abstract: Service-Oriented Architecture (SOA) is a paradigm to implement loosely-coupled, platform-independent distributed software systems using communicating services. Currently, SOA is well-recognized as an established architectural paradigm for distributed systems. Specific design principles for SOA have been recognized as fundamental for implementation in practice. In this article, a systematic literature review is performed to characterize SOA design principles regarding criteria including main domains of application, situations in which SOA is used, which basic design principles are used in practice, how research is evaluated, what benefits of using SOA design principles have been reported, and how SOA applications have been modeled. From 507 articles first selected, 67 were fully read, and 27 were considered of relevance for this research. Interesting results were found from this review. Most studies cited using at least one SOA design principle. Quality of SOA applications has been hardly mentioned. Few studies dealing with the effective application of design principles and how they affect quality are mentioned. SOA Design Principles are presented as key concepts in literature, but are not always shown in case studies as being explicitly applied in practice.

Key words: Service-oriented architecture, systematic literature mapping, service design principles, software quality.

1. Introduction

Several distributed solutions have been proposed for systems integration in past years, with varying degree of success and limitations [1]. SOA (Service-Oriented Architecture) is an attractive technology because it enables reuse of legacy systems as well as possibility of keep using a system instead of developing a complete new version. SOA also facilitates integration of systems running on heterogeneous environments. Legacy systems developed in past decades are known to be inflexible and difficult to maintain for many reasons [2], including their complexity, which makes them difficult to understand, and shortage of personnel who can understand their intricate logic. SOA has emerged with promise of allowing legacy systems to expose their core functionality as services [3].

Currently, SOA is well established as an architectural paradigm for distributed systems [4]. Within SOA, developers can combine and integrate internal legacy software assets with further components, possibly in other networks, with purpose of creating new applications. All those legacy systems that used to operate in

an isolated manner can be defined as services and integrated into new solutions.

Authors of article [5] described a number of research roadmaps on service-oriented computing research. Among the research gaps, they have mentioned *Design principles for engineering service applications* as a research challenge in the near future. Basic principles of software design are important for SOA applications as well. Low coupling between services and high cohesion of services are principles to guarantee that services are self-contained and are well-defined regarding boundaries and interfaces. Therefore, composition of services is facilitated in this case. In this article, we want to investigate further on what was proposed regarding this item of the roadmap. The systematic review proposed in this article brings answers to research questions on what has been researched in academia and industry about web services after the publication of roadmaps presented in [5].

Systematic Literature Reviews (SLR) in Software Engineering have gained importance in recent years [6] [7]-[9] with the purpose of finding empirical evidence on many domains, research areas, and activities of a software's life cycle, including Aspect-Oriented Programming [10], Requirements Engineering [11] and Software Testing [12]. However, specifically for Service-Oriented Architecture, few publications were found. In [13], the authors presented a SLR to identify the state of the art in research on testing in Service Oriented Architectures with dynamic binding. In [14], the presented SLR has as main objective to identify process models for developing SOA applications. Three main objectives of the SLR presented in paper [15] are to assess methods for handling variability in quality attributes of service-based systems, to collect evidence about current research that suggests implications for practice, and to identify open problems and areas for improvement. To the best of our knowledge, there are no specific SLR publications on design principles for SOA, their benefits when applied in practice, and their relation to application quality, as described in this article.

2. Review on SOA Design Principles

According to the SWEBOK (Software Engineering Body of Knowledge), software design principles are key notions, and are considered fundamental to many different software design approaches and concepts. Several design principles for software have been proposed in the literature aiming to improve logic of solution and/or deal with software complexity.

According to [16], a principle of design is defined as a widespread and accepted concept by industry and/or academia. A principle can be compared to a good practice by the fact both propose a means of achieving something based on experience or market acceptance. Erl presents eight specific principles for SOA design: service contract, service loose coupling, service abstraction, service reusability, service autonomy, service statelessnes, service discoverability, and service composability. These design principles are considered fundamental for SOA applications and are widely referenced in literature. A brief description of each principle is highlighted as follows.

- Service contract: services share standardized contracts. A service contract consists of a technical interface or one or more service description documents.
- Service loose coupling: services are loosely coupled. Existence of a service contract, ideally, is decoupled from implementation details and technology.
- Service abstraction: services are designed to limit information in service contract to what is really necessary for each service to be functionally useful to consumers. Information beyond that is published in a service contract and is considered private and should not be made available for creating potential consumers of service.

- Service reusability: services are reusable. Services encapsulate logic that is useful for more than one service request. This logic is generic enough to allow many usage scenarios and can be used for different types of service consumers.
- Service autonomy: services are autonomous. Services have a contract that expresses a well-defined functional threshold which should not involve other services.
- Service statelessnes: services minimize dependence on states. Services are specifically designed to minimize periods during which they exist in a condition of dependence on state information.
- Service discoverability: services are designed to be effectively discovered and interpreted so that discovery, its purpose and capabilities are clearly understood. Service contracts should contain adequate metadata that will be referenced correctly when viewing queries are issued.
- Service composability: services are designed to act as effective composition participants, regardless of size and complexity of composition. Services can be composed to automate business processes of a company.

Article [17] cites most of the design principles proposed by Erl, as well as additional principles, including Service Encapsulation, Service Optimization and Service Relevance. According to the authors, Service encapsulation means many services are consolidated for use under SOA. Often, such services were not planned to be under SOA. Service Optimization means high-quality services are generally preferable to low-quality ones. Finally, Service Relevance is a functionality presented at a granularity recognized by the user as a meaningful service. Nevertheless, the article does not mention who proposed these new principles.

Article [18] mentioned Service Component Modularity, Change Management and Integration as design principles. Article [19] is another work presenting other types of design principles, such as Granularity. Articles [20] and [21] also mentioned service granularity. Granularity is a design principle proposed by Leger and Vogel in 2007 [22]. Article [23] mentioned the design principle Service Oriented Usability. According to [24], usability is a nonfunctional software quality characteristic defined with sub-characteristics. Major sub-characteristics are: understandability, learnability, and operability. Service oriented usability is the grade given by a user of service oriented applications. Within this article, design principles related to usability of services are defined, as for instance, if diagrams are developed for service, if life cycle and lifetime of services are defined, and even if connections to access each service are defined. Article [25] presented another design principle called Business Alignment, meaning the Service provides related set of functionality. For example, a Service related to student registration should not perform functionalities related to Professor.

3. Protocol

Steps for the systematic review include definition of research questions, definition of venues, definition of the search string and testing the search string.

3.1. Research Questions

Research questions addressed in this article are described as follows. The reason for each research question and what is expected from each one is briefly explained after each question.

RQ1: What are the purposes of using SOA?

SOA has been frequently associated with legacy systems [3]. With this question, we want to know most common activities related to using SOA in practice. One can expect that services are used to integrate legacy systems for developing new complex distributed applications. However, we want to know if there are further purposes for using SOA besides integration of systems.

RQ2: What are the most used design principles for SOA applications?

From the list of design principles for SOA, we want to know which ones are most used for developing SOA applications, and also why these are most used. On the other hand, if a design principle is not used at all, then it is of utmost importance to know why.

RQ3: What are the benefits of applying SOA design principles? Are these benefits measured?

With this question, we want to understand not only advantages of SOA design principles, but also understand the expected final quality of SOA applications. This is most useful if one can find metrics to support results, as has been common with other technologies, methods and processes in Software Engineering [26] [27].

RQ4: How are SOA applications evaluated in practice?

Historically, evaluation is an issue in Software Engineering research [28] [29] [30]. We want to know if this is true for SOA applications as well, i.e., if applications developed with SOA paradigm are evaluated in practice, and what are the employed methods for evaluation.

RQ5: How are SOA applications modeled?

We want to know how modeling of SOA applications is performed, and which modeling languages and specific diagrams are used for modeling SOA applications.

RQ6: For which domains are SOA applications developed?

The purpose with this question is to map which domains are most commonly using SOA, and if SOA has been applied only to specific domains. Therefore, we want to know if SOA applications are restricted to specific domains, or are widely applied in industry and academia.

3.2. Search Process

Search was performed in journals and conferences from 2008 to 2014. This choice of dates interval was decided based on two reasons. First reason is because *Design principles for engineering service applications* is a research challenge presented in 2008 [5]. Finding out if this gap has been adequately addressed by researchers in the last seven years is an open question. Another reason is because publications to be searched must be up to date, and more importantly, design principles for SOA were systematized by Erl in his 2007's book [16] (although we have also looked at other design principles recently published).

Search strategies in SLR often involve automatic keyword searches in digital libraries. Venues searched included ACM, ScienceDirect, Springer, and IEEE Xplore Digital Library. The defined search string was ((“service oriented architecture” or “service oriented computing”)) and (“design principles”). The total number of retrieved articles was 507.

According to [31], software engineering digital libraries do not provide good support for identification of relevant research and selection of primary studies. Therefore, a manual selection of articles in relevant venues was also performed. We also searched in IEEE TSE (IEEE Transactions on Software Engineering) and ACM TOSEM (ACM Transactions on Software Engineering and Methodology).

3.3. Inclusion and Exclusion Criteria

A criterion to effectively consider an article for evaluation in this SLR was divided into two steps. First step consisted of analyzing all 507 articles in order to find those that would be the most important ones for this SLR. In this first step, for each article, title, keywords, and abstract were read and catalogued. This is a crucial part of the step which was taken carefully. Thus, the choice was to perform this part independently by both authors. Each author classified each paper, in terms of selection for further steps, as “Definitely”, “Possibly”, or “Not Selected”. For the final list of selected articles, the following rules were applied:

- R1 Article is selected if it has been classified with at least one “Definitely”.
- R2 Article is selected if it has been classified with two “Possibly”.

- R3 Articles classified with two "Not Selected" were obviously not chosen.
- R4 If an article was classified as one "Possibly" and one "Not Selected", then title, abstract and keywords were read again, as well as the conclusion. Then, the article was classified again, and if classification remained the same, it was discarded. Otherwise, one of the first three rules was applied.

These rules were applied by considering the selection process has some qualitative analysis, and it is hard to make it completely deterministic.

After this first step of classification, 67 articles were selected. This number of selected articles can be considered low (about 13%). However, according to article [6], which presents a Systematic Literature Review of Systematic Literature Reviews in Software Engineering, the percentage of selected articles normally varies from a minimum of 1,30% to a maximum of about 67%. Most works selected from 5% to 15% papers. Therefore, the final number of selected papers makes the research relevant.

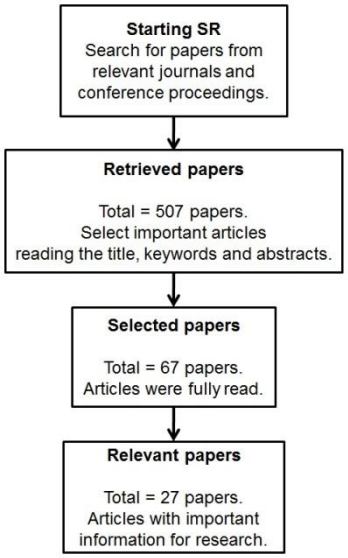

Fig. 1. Paper selection process.

Another reason why the number of selected articles was low is because the word "service" has many meanings, and is a common word even outside the service-oriented domain. However, even when the domain is the right one, most articles were not meaningful for this systematic review because they consider other aspects such as business or infrastructure.

For the second step, all 67 articles were fully read, with the purposes of comprehending main subject and trying to find answers to defined research questions. Articles were considered relevant if they answered at least one of the research questions. In summary, relevant articles considered at least one of the following aspects:

- Article proposes a real application in practice, an experiment, a case study, or industrial cases.
- Article used some kind of modeling for SOA.
- Article mentioned design principles for SOA.
- Article mentioned keywords such as modularity, architecture, coupling, cohesion or granularity.
- Article evaluated influence of a design principle in the final quality of the product.

The overview of the selection process and exclusion is depicted in Fig. 1. Total number of relevant articles, as described in Table 2, was 27 (about 5.5% of initial total number of articles).

Table 1. Overview of Search Results and Selected Papers

Venues	Retrieved Papers	Selected Papers	Relevant Papers
ACM	118	13	7
Science direct	178	24	7
IEEE	63	17	9
Springer	103	6	3
IEEE-TSE	39	4	1
ACM-TOSEM	6	3	0
Total	507	67	27

Table 2. Relevant Papers

Venue	Publication	Venue/Year
ACM	[35]	SPLC/2012
	[48]	ICEGOV/2010
	[49]	GECCO/2013
	[44]	WWW/2009
	[36]	SAC/2008
	[50]	WETSoM/2010
	[51]	JVLC/2014
Science direct	[21]	JSS/2012
	[20]	Env. Model. and Soft./2010
	[52]	Int. Jour. of Medical Inf./2010
	[53]	Expert Sys. App./2012
	[54]	IST/2010
	[43]	Future Gener. Comp. Sys./2009
	[39]	PMC/2014
IEEE	[17]	ICSPS/2010
	[19]	ICIECS/2009
	[37]	ISPA/2010
	[38]	VS-GAMES/2011
	[23]	SOCA/2009
	[25]	SRII Global Conference/2011
	[33]	IEE SCC/2011
	[18]	IEE SCC/2009
	[55]	J-BHI/2014
Springer	[40]	SOCA/2013
	[34]	Computing/2013
	[41]	Business and Inf. Sys. Eng./2011
IEEE-TSE	[42]	IEEE TSE/2011

4. Characterization

Items for characterization of the research on application of Service-Oriented principles, and how SOA is applied in practice, are domain of application, situations in which SOA is applied in practice, which principles are used, how published researches are being validated, found benefits of SOA design principles, and how SOA design has being modeled.

4.1. RQ1 — What Are the Purposes of Using SOA?

In this section, we analyze how SOA has been used in practice. Table 3 shows that SOA has been considered in activities related to refactoring legacy systems, integration of systems with the purpose of producing a new system, and to create new methodologies to develop systems.

Currently, SOA is well-recognized as an established architectural paradigm for distributed systems [4]. Several systems can be connected to exchange information and implement business requirements. Services in SOA are platform independent [32]. This feature is of great advantage because SOA enables integration in heterogeneous environments. Thus, no matter what platform the system has been developed in, it is always possible to propose integration using SOA [1].

Some articles in this review present real applications of using SOA to integrating systems. For example, in [33], a real world application that uses SOA to create a service called AddressManagement is developed.

This service currently retrieves addresses given an address identifier. Other articles present real word applications of SOA through creation of new systems. In these cases, integration of systems originated new systems. For instance, the proposal in [20] is a new software called "The AWARE" project, which overall goal is to provide hydrologists with distributed and reusable tools to monitor and to predict water availability.

Table 3. Use of SOA in Each Article

Article	Refactoring	Integration	New System	New Method
[35]	●	●		
[48]		●	●	
[49]			●	
[44]				●
[36]		●		●
[50]		●		●
[21]	●			
[51]			●	
[20]		●	●	
[52]		●	●	
[53]		●		
[54]		●		
[43]		●		
[39]		●		
[17]	●	●		
[19]		●		
[37]		●	●	
[38]		●	●	
[23]				●
[25]		●	●	
[33]		●		
[18]	●			
[55]	●			
[40]	●			
[34]				●
[41]	●			
[42]	●			

Another important contribution of SOA is to enable reuse of legacy systems. SOA has emerged promising to allow legacy systems to expose their core functionality as services [3].

Some studies in this review presented applications in industry that used SOA to refactoring legacy systems. In one case [18], services are created to update and insert business rules. According to the authors, SOA represents an opportunity to renovate core banking systems in a progressive manner. SOA enables banks to create an open, flexible technology framework, making integration of both internal and external applications relatively simple and allowing for far easier and faster changes to business.

In other studies, SOA is presented with new methodologies. Those articles do not present real applications of SOA, but they propose new methodologies based on SOA. Article [34] proposed a new methodology to teach SOA using concepts of object-oriented programming. Article [23] proposes an approach to measure service oriented usability.

4.2. RQ2 — What Are the Most Used Design Principles for SOA Applications?

Some specific design principles for SOA have been used in the studies. One popular work in specific design principles for SOA is the one proposed in [16]. Some other studies have cited other authors in this area, including [22], [23], [17], [25].

Table 4. Design Principles Used in Each Study

Study	C	LC	A	R	At	S	D	Cm	G	E	O	Rl	U	BA	SM	CM	I	TU
[35]																		
[48]	●	●	●	●	●	●	●	●										
[49]		●				●	●											
[44]	●					●												
[36]																		
[50]				●				●										
[51]								●										
[21]		●	●	●				●	●									
[20]		●		●					●									
[52]		●		●														
[53]							●											
[54]	●	●	●		●		●											
[43]								●										
[39]																		
[17]	●	●	●	●	●		●	●		●	●	●						
[19]		●							●									
[37]																		
[38]																		
[23]													●					
[25]		●		●	●	●								●				
[33]	●		●	●														
[18]				●											●	●	●	
[55]									●									
[40]																		
[34]	●	●	●	●	●	●		●										
[41]									●									
[42]		●	●	●	●			●										

Table 4 presents design principles cited by each study. In this table, the following words were abbreviated as follows. C means Service Contract, LC means Service Loose Coupling, A means Service Abstraction, R means Service Reusability, At means Service Autonomy, S means Service Statelessness, D means Service discoverability, Cm means Service Composability, G means Service Granularity, E means Service Encapsulation, O means Service Optimization, Rl means Service Relevance, U means Service Oriented Usability, BA means Business Alignment, SM means Service Component Modularity, CM means Change Management, I means Integration and TU means Transformative User Experience.

All design principles mentioned in Table 4 were briefly explained in Section 2.

Six out of twenty-seven studies did not consider any design principle. Articles [35], [36], [37], [38], [39] and [40] have not cited using design principles for development of their SOA applications. This does not mean the principles were not considered: it just means they were not mentioned.

Twenty-one out of twenty-seven studies cited they used at least one SOA design principle. Service Loose Coupling and Service Reusability were the most cited principles in studies (both were cited 11 times). In general, design principles described by Erl in [16] - Service Contract, Service Loose Coupling, Service Abstraction, Service Reusability, Service Autonomy, Service Statelessness, Service discoverability and Service Composability - are the most widely applied. Therefore, despite the work of Erl can be considered for business professionals in industry, it is also relevant to academic studies.

Granularity is a design principle proposed in [22] and was considered by 3 studies ([19]-[21]). Service Encapsulation, Service Optimization, Service Relevance, Service Oriented Usability, Business Alignment, Service Component Modularity, Change Management, Integration and Transformative User Experience are less addressed by studies. Each one of these design principle was cited just once.

Some articles (Articles [37], [41], [38], [40]) did not explicitly mention using design principles for

implementation of the case study. Despite this, it is possible to infer that design principles were used during development. For example, in [38] the principle of Service Contract was used because the WSDL standard which specifies service interface is mentioned.

4.3. RQ3 — What Are the Benefits of Applying SOA Design Principles? Are these Benefits Measured?

In this section, we investigate if design principles for SOA provide benefits for final application quality and how these benefits are measured. Few articles mentioned influence of design principles on final software quality.

In [19], authors addressed the influence of design principles in quality of SOA applications. Considered design principles were coupling and granularity. Metrics for measuring coupling and granularity of services were proposed. As for conclusion, the paper explains that, in order to improve quality of SOA applications, it is critical to obtain low values in coupling metrics and high values in granularity metrics.

Author of article [25] states Services that follow design principles are robust to changes and are largely reusable in multiple scenarios but in similar domains. Based on this, the article proposes a formal, rigorous approach to check adherence of design principles in the solution. Considered design principles were Business Alignment, Coupling, Reusable, Autonomous and Stateless. This approach will help designers to validate if Services follow principles at design time.

Table 5. Validation in Each Study

Study	Case Study	Controlled Experiment	Without Validation
[35]	●		
[48]	●		
[49]	●		
[44]			
[36]	●		
[50]	●		
[51]	●		
[21]	●		
[20]	●		
[52]	●		
[53]	●		
[54]	●		
[43]			●
[39]	●		
[17]	●		
[19]	●		
[37]	●		
[38]	●		
[23]			●
[55]	●		
[25]	●		
[33]	●		
[18]	●		
[40]	●		
[34]			●
[41]	●		
[42]	●	●	

4.4. RQ4 — How Are SOA Applications Evaluated in Practice?

Most studies were validated by using case studies. Only one article was evaluated with a controlled Experiment (Table 5).

Two types of validation were performed in [42]. First one is a case study: a SOA software developed specifically for this study, based on an existing service-oriented Academic Management System. Second

step on this study is to conduct a controlled experiment to evaluate proposed coupling metrics.

Three studies - [43], [23], [34] - were proposed without any kind of validation.

Article [44] produced a quantitative evaluation by comparing the degree of coupling implied by different Web services technologies: RESTful HTTP, RPC over HTTP, and WS-*.

4.5. RQ5 - How Are SOA Applications Modeled?

Some issues about modeling SOA are addressed in this review, such as how modeling is performed, and which modeling languages and specific diagrams are used for modeling SOA applications. Table 6 shows which modeling diagrams have been used to design SOA applications. The reason why not all 27 studies appear in Table 6 is because only studies that explicitly applied modeling diagrams are presented in this Table.

In Table 6, words had to be abbreviated due to lack of space. FC means Flowcharts, DF means Data Flow, FSG means Functionality and Service Graph (a graph defined to identify services based on an aggregation or disaggregation of functions or functionality). With respect to UML diagrams, Activ means Activity Diagram, Class means Class Diagram, Collabo means Collaboration Diagram, Comp means Components Diagram and Seq means Sequence Diagram.

Another study [36] represents services graphically, but the diagrams defined by authors did not follow any standard. Due to lack of standardization, the understanding of the diagrams is difficult.

Nine studies have used some kind of diagram to model the developed system. Most studies, 18 out of 27, did not mention any kind of modeling language.

Table 6. Modeling Diagrams Used in Each Study

				UML Diagrams				
Study	DF	FC	FSG	Activ	Class	Collabo	Comp	Seq
[21]	●							
[20]				●				
[52]							●	
[54]								●
[43]		●						
[25]								●
[41]			●					
[42]	●	●			●	●		●
[53]								●

SOA applications have been modeled with different diagrams. Each software designer represents the system and services which must be developed in whatever way is most appropriate. However, this represents a lack of standardization in this area. In addition, the fact most studies did not mention using diagrams can mean the description of service functionalities is performed in natural language, which can lead to problems related to misunderstandings, inconsistencies, and ambiguities [45], [46]. In this context, it is possible to reflect on some questions for future research. Can SOA applications be fully modeled using general purpose diagrams, such as UML diagrams? Does Natural language has been used due to lack of adequate diagrams?

Surprisingly, SoaML [47], a modeling language specifically defined for designing SOA applications, was not mentioned in this review. Reasons for this are not yet clear, and need to be investigated in future research. One possibility is the novelty of the language, officially released on March 2012 (even tough beta versions were published in 2009).

4.6. RQ6 — For Which Domains Are SOA Applications Developed?

Articles found in this SLR present diversity in terms of fields of application of SOA. Case studies of each

article are presented in Table 3. This listing has been performed to show which application domains are used in the case studies. It is interesting to note that SOA has been applied in different domains, and the application of SOA is comprehensive for various domains. In Table 3, one can analyze that SOA has been applied from academic applications to military, healthcare and infrastructure systems.

Table 7. Domain Application of Each Article

Article	Domain	Comments
[35]	Military system	United States Army Program application to military live training community.
[48]	Public Sector	E-services (e.g., administrative, judicial, financial) for a customer perform the whole process.
[49]	Evolutionary Computation	Design and implementation of services for Evolutionary Computation.
[44]	Not-described	-
[36]	Purchase Order	Integration of the ordering application of a customer.
[50]	Healthcare application	Healthcare processes related with Visiting Preparation.
[51]	Personal Information Spaces application	a platform that allows end users, who are not necessarily experts of technologies, to compose Personal Information Spaces
[21]	Hospital and Belgian banking	Real example
[20]	Geospatial Services	Discovery, access, processing and visualization of geospatial data
[52]	Clinical decision support	Healthcare information system
[53]	Geospatial data	To find and access geospatial data over Internet using a single tool
[54]	Purchase products	Simulation of consumers purchasing products from a large firm
[43]	Not-described	-
[39]	Automotive Application	A cooperative convoy telematics system that allows interaction and collaboration between automotive vehicles.
[17]	Remote Collaborative Experiment Systems	Enables storing and sharing data about scientific experiments
[19]	Claim Approval	Create, pay and approve claims
[37]	System for Financial Analysis	Pricing and data/information service system for financial analysis of securities
[38]	System for evaluation of disaster management	Collect, process, visualize and interpret geospatial data to intelligent support decision making
[23]	Not-described	-
[25]	Academic System	Student Course Registration System (not implemented).
[33]	Address Management	Perform retrieval of matching addresses given an address identifier.
[18]	Asian bank	Real application
[55]	Healthcare application	system for drug prescription, administration, and registration
[40]	Student information system	Real system
[34]	Not-described	-
[41]	Financial Service Provider	Loan division of major German financial service provider.
[42]	Academic Management System	Software system developed specifically for this study

5. Conclusion

According to the presented SLR, SOA has been applied in various areas, including financial systems, manufacturing of goods, healthcare information systems, research-oriented applications in engineering, and academic software applications. Therefore, it is safe to assume that SOA has been widely applied in projects both in academia and industry.

Specific design principles for SOA were proposed in literature with the purpose of being crucial guides on how to create applications based on services. However, few studies dealing with

applying design principles and how they affect quality are mentioned. Design principles guide about how services can be reused, but the high reusability of an application is to be defined by good decisions taken by developers, which depends on knowledge and experience. This is true as well regarding granularity.

Although SOA design principles are considered fundamental for SOA applications and are widely referenced in literature, authors of six out of twenty-seven studies have not explicitly mentioned whether they have used SOA design principles. Twenty-one out of twenty-seven studies cited using at least one SOA design principle. Sixteen studies cited using two or more design principles. Design Principles are presented as key concepts, but are not always shown in case studies as being explicitly used and how they are used in practice. Design principles are proposed as key concepts to help developing good design solutions. However, only two articles mentioned influence of design principles on final software quality. Quality of SOA applications has been poorly mentioned as described in this article.

Even though a specific modeling language for SOA has been proposed, the SoaML modeling language was not mentioned in this review. The reasons for this result are not clear, and can only be speculated. One possibility refers to the novelty of the language. In addition, UML is well-known in academia and software industry, which makes it a suitable choice for most authors.

As for limitations of this work, the number of selected articles can be considered a threat to validity. Although most systematic literature reviews selected from 5% to 15% papers, and the one presented in this article selected approximately 6%, the final number of selected papers, 27, could be higher in future works. Another threat to validity is that results are based on information written in published papers only, not on interviews with authors. Therefore, even when a design principle is not mentioned in a study, it is possible the principle was used but the authors did not mention. Another example is the choice of modeling language. Even in articles which did not mention the use of modeling languages, one cannot infer the authors of respective articles did not use one.

Acknowledgements

We would like to thank the Brazilian research agency CNPq (grant 445500/2014-0).

References

[1] Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2010). *Web Services: Concepts, Architectures and Applications*. Springer Publishing Company, Incorporated.

[2] Bennett, K. (1995). Legacy systems: Coping with success. *IEEE Software*, *12(1)*,19–23.

[3] Khadka, R., Reijnders, G., Saeidi, A., Jansen, S., & Hage, J. (2011). A method engineering based legacy to SOA migration method. *Proceedings of the 27th IEEE International Conference on Software Maintenance* (pp. 163–172).

[4] Goeb, A., & Lochmann, K. (2011). A software qquality model for SOA. *Proceedings of the 8th International Workshop on Software Quality* (pp. 18–25).

[5] Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2008). Service-oriented computing: A rresearch roadmap. *International Journal of Cooperative Information System*, *17(2)*, 223–255.

[6] Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering — A systematic literature review. *Information and Software Technology*, *51(1)*, 7–15.

[7] Kitchenham, B., & Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and Software Technology*, *55(12)*, 2049–2075.

[8] Zhang, H., & Babar, M. A. (2013). Systematic reviews in software engineering: An empirical

investigation. *Information and Software Technology*, *55(7)*, 1341–1354.

[9] Wohlin, C., & Prikladnicki, R. (2013). Systematic literature reviews in software engineering. *Information and Software Technology*, *55(6)*, 919–920.

[10] Ali, M. S., Babar, M. A., Chen, L., & Stol, K.-J. (2010). A systematic review of comparative evidence of aspect-oriented programming. *Information and Software Technology*, *52(9)*, 871–887.

[11] Pacheco, C., & Garcia, I. (2012). A systematic literature review of stakeholder identification methods in requirements elicitation. *Journal of Systems and Software*, *85(9)*, 2171–2181.

[12] Engstrom, E., Runeson, P., & Skoglund, M. (2010). A systematic review on regression test selection techniques. *Information and Software Technology*, *52(1)*, 14–30.

[13] Palacios, M., Garcia-Fanjul, J., & Tuya, J. (2011). Testing in service oriented architectures with dynamic binding: A mapping study. *Information and Software Technology*, *53(3)*,171–189.

[14] Lane, S., & Richardson, I. (2011). Process models for service-based applications: A systematic literature review. *Information and Software Technology*, *53(5)*, 424 – 439.

[15] Mahdavi-Hezavehi, S., Galster, M., & Avgeriou, P. (2013). Variability in quality attributes of service-based software systems: A systematic literature review. *Information and Software Technology*, *55(2)*, 320 – 343.

[16] Erl, T. (2007). SOA principles of service design*. The Prentice Hall Service-Oriented Computing Series from Thomas Erl.*

[17] Xing, Y., & Yao, E. (2010). Remote collaborative experiments based on service-oriented architecture. *Proceedings of the International Conference on Signal Processing Systems (ICSPS)* (pp. 605–608).

[18] Liu, R., Wu, F., Patnaik, Y., & Kumaran, S. (2009). Business entities: An SOA approach to progressive core banking renovation. *Proceedings of the IEEE International Conference on Services Computing* (pp. 466–473).

[19] Xiao-Jun, W. (2010). Metrics for evaluating coupling and service granularity in service oriented architecture. *Proceedings of the International Conference on Information Engineering and Computer Science* (pp. 1-4)

[20] Granell, C., D´ıaz, L., & Gould, M. (2010). Service-oriented applications for environmental models: Reusable geospatial services. *Environmental Modelling and Software*, *25(2)*,182–198.

[21] Monsieur, G., Snoeck, M., & Lemahieu, W. (2012). Managing data dependencies in service compositions. *Journal of Systems and Software*, *85(11)*, 2604–2628.

[22] Legner, C., & Vogel, T. (2007). Design principles for B2B services — An evaluation of two alternative service designs. *Proceedings of the IEEE International Conference on Services Computing* (pp. 372–379).

[23] Liu, L. L.. (2009). Design principles and measurable service oriented usability. *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications* (pp. 1–4).

[24] ISO/IEC. Software engineering — Product quality, ISO/IEC 9126-1. *Technical Report, International Organization for Standardization*.

[25] Kannan, K., Bhamidipaty, A., & N. C. Narendra. (2011). Design time validation of service orientation principles using design diagrams. *Proceedings of the Annual (Services Research and Innovation Institure) SRII Global Conference*.

[26] Smite, D., Wohlin, C., Gorschek, T., & Feldt, R. (2010). Empirical evidence in global software engineering: A systematic review. *Empirical Software Engineering*, *15(1)*, 91–118.

[27] Harman, M., Mansouri, S. A., & Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, *45(1)*.

[28] Glass, R. L., Vessey, I., & Ramesh, V. (2002). Research in software engineering: An analysis of the literature. *Information and Software Technology*, *44(8)*, 491–506.

[29] Shaw, M. (2002). What makes good research in software engineering. *Int. Journal of Software Tools for Technology Transfer*.

[30] Glass, R. L., V. Ramesh, & Vessey, I. (2004). An analysis of research in computing disciplines. *Communications of the ACM*, *47(6)*, 89–94.

[31] Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80(4)*, 571–583.

[32] Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. *Proceedings of the Fourth Int. Conf. on Web Information Systems Engineering*.

[33] Jacobsen, H.-A., Wang, Y., & Hung, P. (2011). SATE-service BOUNDARY and abstraction threshold estimation for efficient services design.

[34] Stubbings, G., & Polovina, S. (2013). Levering object-oriented knowledge for service-oriented proficiency — Enhancing service capability in developers. *Computing*, *95,* 817–835.

[35] Lanman, J. T., Darbin, R., Rivera, J., Clements, P. C., & Krueger, C. W. (2013). The challenges of applying service orientation to the U.S. army's live training software product line.

[36] Dirgahayu, T., Quartel, D., & Sinderen, M. V. (2008). Designing interaction behaviour in service-oriented enterprise application integration. *Proceedings of the 2008 ACM Symposium on Applied Computing.*

[37] King, G.-H., Cai, Z.-Y., Lu, Y.-Y., Wu, J.-J., Shih, H.-P., & Chang, C.-R. (2010). A high performance multi-user service system for financial analytics based on web service and GPU computation. *Proceedings of the International Symposium on Parallel and Distributed Processing with Applications*.

[38] Vescoukis, V., & Dulamis, N. D. (2011). Disaster management evaluation and recommendation. *Proceedings of the Conference in Games and Virtual Worlds for Serious Applications* (pp. 244–249).

[39] Kabir, M. A., Han, J., & Colman, A. W. (2014). Sociotelematics: Harnessing social interaction relationships in developing automotive applications. *Pervasive and Mobile Computing*, *14*, 129–146.
Baghdadi, Y., & Al-Bulushi, W. (2015). A guidance process to modernize legacy applications for SOA. *Journal Service Oriented Computing and Applications*, *9(1)*, 41-58.

[40] Krammer, A., Heinrich, B., Henneberger, M., & Lautenbacher, F. (2011). Granularity of services — An economic analysis. business and information systems engineering, *3(6)*, 345–358.

[41] Perepletchikov, M., & Ryan, C. (2011). A controlled experiment for evaluating the impact of coupling on the maintainability of service-oriented software. *IEEE Transactions on Software Engineering*, *37(4)*, 449–465.

[42] Wu, B., Chi, C.-H., Chen, Z., Gu, M., & Sun, J. (2009). Workflow-based resource allocation to optimize overall performance of composite services. *Future Generation Computer Systems*, *25(3)*, 199–212.

[43] Pautasso, C., & Wilde, E. (2009). Why is the web loosely coupled? A multi-faceted metric for service design. *Proceedings of the 18th International Conference on World Wide Web* (pp. 911–920).

[44] Kamsties, E. (2005). Understanding ambiguity in requirements engineering. *Engineering and Managing Software Requirements*.

[45] Soares, M. S., Vrancken, J., & Verbraeck, A. (2011). User requirements modeling and analysis of software-intensive systems. *Journal of Systems and Software*, *84(2)*, 328– 339.

[46] OMG. Service oriented architecture modeling language (SoaML) specification. Retrieved 2012, from http://www.omg.org/spec/SoaML/1.0/

[47] Cellary, W., & Strykowski, S. (2009). E-government based on cloud computing and service-oriented architecture. *Proceedings of the ACM International Conference Proceeding Series*.

[48] García-Sánchez, P., Arenas, M. I. G., Mora, A. M., Castillo, P. A., Fernandes, C., Cuevas, P. D. L., Romero, G., González, J., & Merelo, J. J. Developing services in a service oriented architecture for evolutionary

algorithms. *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*.

[49] Khoshkbarforoushha, A., Jamshidi, P., & Shams, F. (2010). A metric for composite service reusability analysis. *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics*.

[50] Ardito, C., Costabile, M. F., Desolda, G., Lanzilotti, R., Matera, M., Piccinno, A., & Picozzi, M. (2014). User-driven visual composition of service-based interactive spaces. J*ournal of Visual Languages and Computing, 25(4)*, 278–296.

[51] Cho, I., Kim, J., Kim, J., Kim, H. Y., & Kim, Y. (2010). Design and implementation of a standards-based interoperable clinical decision support architecture in the context of the Korean EHR. *International Journal of Medical Informatics*, *79(9)*, 611–622.

[52] Tian, Y., & Huang, M. (2012). Enhance discovery and retrieval of geospatial data using SOA and semantic web technologies. *Expert Systems with Applications*, *39(16)*,12522–12535.

[53] Yamany, H. F. E., Capretz, M. A. M., & Allison, D. S. (2010). Intelligent security and access control framework for service-oriented architecture. *Information and Software Technology*, *52(2)*, 220–236.

[54] Bianchi, L., Paganelli, F., Pettenati, M. C., Turchi, S., Ciofi, L., Iadanza, E., & Giuli, D. (2014). Design of a restful web information system for drug prescription and administration. *IEEE J. Biomedical and Health Informatics*, *18(3)*, 885–895.

[55] (2004). *Software Engineering Body of Knowledge (SWEBOK).* IEEE Computer Society. Angela Burgess, EUA.

Joyce M. S. França received her BS in computer science in 2011 and her MS in computer science in 2013, both from Federal University of Uberlândia in Brazil, and is currently a PhD student with focus on software engineering from Federal University of Uberlândia, in Brazil.

She has been working as a software developer, researcher and then an assistant professor since 2013. Currently she is an assistant professor at Federal Institute of Education, Science and Technology of Norte de Minas Gerais, Brazil. Her research interests include software architecture, service-oriented architectures and software quality. MSc. França has worked in some research projects and has published 7 articles in scientific conferences.

Michel S. Soares received his BS in computer science in 2000 from Federal University of São Carlos and his MS in computer science in 2004 from Federal University of Uberlândia, both in Brazil, and a PhD with focus on software engineering in 2010 from Delft University of Technology, in The Netherlands.

He has been working as a software developer, researcher and then an assistant professor since 2000. Currently he is an assistant professor at Federal University of Sergipe, Brazil. His research interests include software architecture, requirements engineering, service-oriented architectures and software quality.

Dr. Soares has published over 70 articles in scientific journals, conferences and books.