

A Systematic Literature Review for Service-Oriented Architecture and Agile Development

James Taylor Faria Chaves(✉) and Sergio Antônio Andrade de Freitas(✉)

University of Brasilia, Darcy Ribeiro Campus, Brasília, DF, Brazil
jameschaves@gmail.com, sergiofreitas@unb.br

Abstract. This research presents a Systematic Literature Review (SLR) involving Agile Methods and Service-Oriented Architecture (SOA). Is this combination of solutions capable of handling rapid changes in the environment, business objectives, and requirements? To answer this question and give confidence to this review, a rigorous protocol was followed and relevant studies were selected, data were extracted and analyzed. Some studies propose a mature solution, although they do not present data that are more consistent and capable of being evaluated. Other studies are superficial. In addition, few studies have gone through the protocol. So the main conclusion is that combining SOA and Agile methods can be a good way to solve the problem, but more studies are needed.

Keywords: Agile methods · Service-Oriented Architecture (SOA) · Systematic Literature Review (SLR) · Software engineering

1 Introduction

Service-Oriented Architecture (SOA) and Agile development methods. This is the combined solution that some researchers and practitioners are using to address the need for rapid change in environment, business goals, requirements, and even customer interests. This Systematic Literature Review (SLR) evaluated and interpreted relevant searches that address this solution to see if that combination is able to handle the problem.

Through a rigorous protocol, some studies were selected and analyzed from three points of view. It was evaluated the solutions proposed by the researchers and how they addressed the problems involving SOA based applications with agile development. It was also evaluated how the concepts of the agile methodology were used and what SOA principles were addressed. Information about the distribution of publications over the years and the type of paper sources have also been extracted to assess the importance of the subject in the area of software engineering.

Despite the interest of the subject, only 18 studies passed the review protocol. Also, none of them presented a consistent evaluation of the results. However, they

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11623, pp. 120–135, 2019.
https://doi.org/10.1007/978-3-030-24308-1_11

demonstrated the importance of the theme. All studies presented a problem to be solved, and they all concluded that the SOA and Agile solution would meet, at least in part, the issues raised. Therefore, due to the small number of studies and the needs presented, one of the conclusions of this review is that more studies need to be done to better understand the phenomena and their implications.

Section 2 provides an overview of SLR, SOA, and Agile Methods concepts. Section 3 explains the research methods used to carry out this review. Section 4 presents the results of the extraction data from the accepted studies. Section 5 addresses the research questions of this review and Sect. 6 presents the final conclusions.

2 Background

2.1 Systematic Literature Review

Systematic Literature Review (SLR) is an instrument used for synthesizing evidence in some research area [29]. It is a rigorous methodology of review of research results. Its purpose is to aggregate all existing evidence on a research question. The origin of SLR was in medicine researchers [22]. Kitchenham [21] presented a guideline for performing SLR in Software Engineering that was derived from existing guidelines used by medical researches. According to Kitchenham [21]:

"A systematic literature review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. Individual studies contributing to a systematic review are called primary studies; a systematic review is a form a secondary study."

The SLR has more scientific value than a simple literature review due to its methodological rigor. The reasons for performing a SLR are to summarize the existing evidence concerning a treatment or technology, to identify any gaps in current research in order to suggest areas for further investigation and to provide a framework/background in order to appropriately position new research activities [21].

2.2 Agile Development

A light method for software development can be understood as a method that the needs and goals of the stakeholders drive the development and enable the rapid and flexible development of software. The processes and tools itself are less important than deliverables, people, developers and stakeholders, communication, and the ability to adapt to change. This type of method has to be iterative, incremental, cooperative and adaptable to changes in environments and requirements. The search for this type of software development method led to the "agile methods" that were stated in the Agile Manifesto, published in 2001 [2] where seventeen developers and consultants representatives of light methods for software development met to discuss better ways of developing software.

There are many methods of software development that claim to be agile such as SCRUM, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Internet Speed Development (ISD), Lean Software Development (LSD), Crystal Clear, Extreme Programming (XP), Feature-Driven Development (FDD), Pragmatic Programming (PP), and so on [13,16,18,36,37]. Even Rational Unified Process (RUP was used by some authors as an agile method [19,30]. Each has its own phases and life cycles, and some methods do not even well define the phases of development.

2.3 Service-Oriented Architecture

Organizations have their own business goals and they need software systems to satisfy those goals, sometimes with crosscutting stakeholders concerns. One of the key challenges to building systems that meet the goals of large numbers of stakeholders is integrating and reusing existing systems, while adding new functionality to support the business processes and to respond to changes in the business. Software Architecture is a bridge between those goals and the final resulting system, and comprehends the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both [12,24]. In the ending of the 20th century, new approaches to system architecture have emerged which aim to structure a complex system around units of capability, called services [5].

This is the context in which Service-Oriented Architecture (SOA) is placed, that has emerged as a widely accepted solution to this challenge allowing a homogeneous enterprise-wide solution satisfying objectives that include an easy and flexible integration with legacy systems and simplifying business processes. Software architecture can be composed by several design patterns simultaneously to reach its goal and SOA is a paradigm, a design pattern, not a complete architecture. This design pattern was already being used since the beginning of 1990s, mainly in Simple Object Access Protocol (SOAP) [14,41] form. In the mid 2000s, the Representational State Transfer (REST) [33,41] form began to spread and today its hard do say witch proportion news applications are made in one or other form, existing intermediary forms, between both [41]. SOA is a sort of architecture that uses standards-based infrastructure to forge large-scale systems out of loosely coupled, inter-operable services, distributed, invocable, publishable and business oriented that can act like providers, consumers or Information Technology (IT) elements that join providers and consumers. SOA can create systems-of-systems by mapping existing systems into services, then orchestrating communication between the services. New functionality can be created by either adding new services or modifying communication among existing services with low costs, innovators services to clients, agile adaption, and reaction to opportunity and weakness competitiveness [5,6,17,26,35].

SOA can also be understood as an architectural style and uses services as building blocks to embrace changes in the business environment by composing of services and creating composite services already existing with applications in a technology heterogeneous environment. The business and technical processes

are implemented as services and each service represents a particular functionality that maps explicitly to a step in a business process [3]. In this way, service can be understood as any task or function provided by an organization, aligned to business, well defined and isolated from other tasks (autonomy principle) [14] and Service orientation help organizations consistently deliver sustainable business value, with increased agility and cost effectiveness, in line with changing business needs [4]. In general, the basic Service Consumer software is simply a browser or browser-like "thin client", with little or no application software stored locally.

Erl [14] presents the eight principles that a service-oriented solution should follow:

1. Standardized Service Contract: a service should express its purpose and capacities through a contract;
2. Service Loose Coupling: the dependencies among the service's contract, implementation and consumers should be minimal;
3. Service Abstraction: a service should hide its implementation details to keep its consumers loosely coupled;
4. Service Reusability: services should be corporate resources, agnostic to functional contexts;
5. Service Autonomy: a service should have control over its environment and resources, while remaining isolated from other services;
6. Service Statelessness: services should only hold state information when necessary in order to not compromise its availability or scalability;
7. Service Discoverability: services should be easily discovered and understood to foster reuse;
8. Service Composability: it should be possible to create services from the composition of other services in order to produce sophisticated solutions according to business needs requirements and fostering reuse of existing assets

Service-Oriented Architecture (SOA) is not just a set of services connected together. The services consumers and the services providers are connected through an infrastructure responsible to implementing many of the principles of SOA, such as discoverability and composability, as well as responsible for transforming data according to standardized service contracts and dealing with security issues [14].

3 Research Method

In this Systematic Literature Review (SLR) it was used the guidelines proposed by Kitchenham [21] and Kitchenham and Charters [23] and thus, the next steps where followed:

1. Planning the review:
   (a) Identification of the need for a review
   (b) Developing a review protocol
2. Conducting the review:

(a) Identification of research
(b) Selection of primary studies
(c) Study quality assessment
(d) Data extraction & monitoring
(e) Data synthesis
3. Reporting the review

A review protocol is a written plan that is completed before the review begins and it describes every aspect of the review from the background, the rationale for the survey, to the final report. A predefined protocol reduce the possibility researcher bias and provides a means by which the review itself can be repeated or updated at a later date to include subsequent publications [21,25]. According to Kitchenham [21] the background is the rationale for the survey, explain the need for the review. In this paper the background is showed in the Sect. 2. In the following subsections, it is presented the elements of the review according to the developed review protocol and some additional planning information.

3.1 Research Questions

The primary and secondary research questions elaborated for this study are:

1. Primary Research Question (RQ01): What Agile solutions are proposed for developing Service-Oriented Architecture (SOA) based Applications?
   - Population: all roles in software development process;
   - Intervention: agile Processes for development SOA based applications.
   - Control: processes for traditional software development;
   - Outcomes: set of processes, frameworks, models, and techniques for developing SOA based applications with Agile methods.
2. Research Question (RQ02): What phases of the agile life cycle are the studies covering?
3. Research Question (RQ03): What common principles of Service-Oriented Architecture (SOA) are the studies covering?

The RQ02 and RQ03 research questions complement the RQ01 research question. The research question RQ01 tries to answer what kind of contribution the studies bring, process, method, tool, etc. In addition, the RQ02 research question seeks to identify at which phases of the life cycle is the focus of the studies and the RQ03 research question is concerned with identifying how common principles of SOA are addressed in the studies.

3.2 Search String and Data Sources

To compose the search string the main words and synonyms were identified. Table 1 describes these synonyms. At first, other synonyms were tried, such as Extreme Programming (XP), SCRUM, Lean Software Development (LSD), and Crystal, but these synonyms did not add much, and sometimes brought undesirable works. In addition to the selected synonyms, the words “architecture”,

Table 1. Search synonyms

Agile Development	Service-oriented Architecture
Agile Method	SOA
Agile Programming	Service-Based Architecture (SBA)
Agile Process	Service-Oriented Computing (SOC)
Agile Practice	Webservice
Agile Requirement	Representational State Transfer (REST)
Agile Technique	Simple Object Access Protocol (SOAP) Service-based

"computing", "system", and "application" were added to limit the result only to the type of study desired.

All of the used electronic data sources work with a well know structure to ask about electronic papers, using the operator 'OR' to include synonyms for each search term, and the operator 'AND' to link together each set of synonyms. Thus, the search string used was:

((*SOA* OR *SBA* OR *SOC* OR *webservice** OR *"web service"* OR *"web services"* OR *REST* OR *SOAP* OR ((*"software-oriented"* OR *"Software oriented"*) AND (*architecture* OR *computing*)) OR ((*"service-based"* OR *"service based"*) AND (*application** OR *system*))) AND (*"AGILE DEVELOPMENT"* OR *"AGILE MANUFACTURING"* OR *"AGILE METHOD"* OR *"AGILE PROGRAMMING"* OR *"AGILE PROCESS"* OR *"AGILE PRACTICE"* OR *"AGILE REQUIREMENT"* OR *"AGILE TECHNIQUE"*))

Only the electronic data sources depicted in Table 2 were used. Each data source has its own peculiarity to do the search, so the search string above needed to be adapted for each case, but without losing its essence.

Table 2. Data sources

Source	URL
IEEE	https://ieeexplore.ieee.org
ACM	https://dl.acm.org/
Web of Science	https://www.isiknowledge.com
Springer	https://link.springer.com/
Science Direct	https://www.sciencedirect.com
Scopus	https://www.scopus.com

3.3 Study Selection and Data Extraction

Table 3 depicts the selection criteria used in this review. Some criteria are "Automatically identified" and all electronic data sources used in this review provide advanced search tools that allow filtering of automatic criteria. Even though the

Table 3. Study selection criteria

Studies' inclusion criteria
- Automatically identified
- Published and available in electronic data sources
- Contain search keywords in the title, abstract or author keywords
- In English
- All publication years
- Personally identified
- About SOA development using Agile Development Method
- Have to be more than four pages
Studies' exclusion criteria
- Studies that cite SOA and Agile although it isn't the main purposes to explain about development SBA using agile
- Papers which are obviously not related to the research questions in this protocol
- Papers that are application domain specific

original data source search tool does not provide all the necessary features, some parameters are easy to identify in standalone tools, such as an electronic data sheet.

Other criteria for inclusion and exclusion of studies, not "Automatically identified", were identified through a careful reading of the authors title, abstract and keywords. Some studies were excluded late, in the data extraction phase, during the full reading. This was because a conservative criterion was used in this selection phase: *'in case of doubt the study remains'*.

The State of the Art through Systematic Review (StArt)[1] tool was used to aid in data extraction. This tool helps to organize all the work and its two main features are that it follows the protocol proposed by Kitchenham [21] and its local database is a SQLite database[2], which allowed to directly access the tables and work with the data in a Structured Query Language (SQL)[3] way.

3.4 Data Synthesis Strategy

Viewing all collected data is not the best way to understand the data. For the synthesis, the data was stratified the by year, paper sources, and type of solution presented. Table 4 shows the fields used to assist in extracting data.

To classify the solutions found in the studies, they were grouped in phases, or processes, of the software development life-cycle, that is, in which stage of the development life cycle the proposed solution fits best. Proposed by Kitchenham and Charters [23], the reciprocal translation method [28] was used. The processes found in the studies were classified into groups of related processes. But first, it

[1] http://lapes.dc.ufscar.br/tools/start_tool.
[2] https://www.sqlite.org.
[3] https://en.wikipedia.org/wiki/SQL.

Table 4. Data extraction fields

Data field	Purpose
Reviewer	Name of reviewer
Title	Name of study
Authors	Study authors
Publication source	Where the paper was published, name of journal, conference, etc.
Publication type	Is the paper a conference paper, journal paper, etc.?
Year of publication	When study was published
Research question	Research question(s) of the study
Study focus	Primary objective of the study
Processes	Process described in the study
Findings/conclusions	Main conclusions from the study
IS valid	Was it a valid study?

was necessary to choose a specific development life-cycle to be used by a pattern. The options to chose were the life-cycles related to SOA or those related to Agile development.

Defining a life-cycle for agile software development is more difficult as agile in essence does not follow a rigorous process, as it can be seen in the Manifesto for agile software development [2]. By the manifesto it is possible to see that processes, tools, comprehensive documentation, negotiation of contracts, and following a plan is not the most important. Thus the development meta-model SOA proposed by Lane and Richardson [25] has been chosen. In this meta-model, the phases are 'Analysis and Design (AD)', 'Construction and Testing (CT)', 'Deployment and Provisioning (DP)' and 'Implementation and Monitoring (IM)'.

4 Data Extraction Results

After the planning phase and with the protocol at hand, it was time to execute the review, select the studies and extract and analyze the data. Data extraction occurred during August 2018. Each data source has its own search tools with its own characteristics. The search query had to be adapted to take full advantage of the potential of each tool.

In the first phase, 146 studies were selected, already discounting the duplicate papers, that is, papers that were available in more than one data source. All abstracts were read and a total of 124 studies were rejected. The main reason for the rejection was that while the studies cited SOA and Agile, the main goal was not to address the development of SOA based application using Agile methods. Then, 22 studies have gone into the data extraction phase.

In data extraction phase, the full text of the 22 studies was read and the extraction form based in Table 4 was filled. At this phase, 4 papers were rejected. Although these papers addressed SOA and Agile methods, they did not address

any of the three research questions. Then, 18 studies remained. They are the objects of this review, listed in Table 5.

Table 6 shows the solutions of each study bound to its respective phase in the meta-model after the process of reciprocal translation. Most of the solutions were linked in the 'Analysis and Design' phase of the meta-model. In this type of solution, the authors pay special attention to the definition of the system architecture and its components, linked to the business processes. The 'Construction and Test' phase of the meta-model also received a lot of attention from the researchers, especially regarding the development process itself. Two studies addressed a solution to governance and one to security. The 'Deployment and Provisioning' phase received only 2 studies. The 'Execution and Monitoring' phase did not receive any solution. This demonstrated that most authors were more interested in delivering products and less on how to maintain them. All solutions have presented concepts based on agile methods in general or on specific development methods, such as Rational Unified Process (RUP) or Extreme Programming (XP).

Although the research has not limited the origin of the paper, no book or book chapter has been selected. Most of the paper came from the Conference

Table 5. Categorical data extracted from selected studies

Study	Source	Year	Solution	Meta-model	Q1	Q2	Q3
[1]	CP	2014	Framework/Model	AD/CT/DP	Y	Y	
[7]	SP	2013	Approach/Other	CT	Y	Y	Y
[8]	CP	2012	Approach/Framework	AD/CT	Y		
[9]	WP	2013	Framework	CT	Y		
[10]	JP	2017	Model	AD/CT/DP	Y		
[11]	JP	2010	Process	CT	Y	Y	
[12]	CP	2007	Model/Process	AD	Y		Y
[15]	CP	2011	Framework	CT	Y		
[19]	JP	2003	Approach	AD/CT	Y		
[20]	JP	2008	Approach	AD	Y		
[24]	CP	2005	Other	AD	Y		
[30]	CP	2013	Process	AD/CT	Y	Y	
[31]	CP	2008	Framework	AD/CT	Y		
[32]	CP	2010	Model	CT	Y		Y
[34]	JP	2012	Approach	AD	Y		
[38]	CP	2017	Approach	AD/CT	Y		
[39]	CP	2010	Model	AD	Y		
[40]	SP	2016	Other	AD	Y		Y

CP - Conference Paper/JP - Journal Paper
SP - Symposium Paper/WP - Workshop Paper

Table 6. Reciprocal translation detail

Analysis and Design (AD)	
Framework to design, implementation and deployment of SOA*	[1]
Solution to defining architecture	
Analysis, Architecture Elaboration, Granularity Identification	[10]
SOA Based Model Driven Rapid Development Architecture	[39]
BPMN model	[8]
Composition and orchestration of services	[20]
Establish a Core SOA Architecture	[34]
Architecture Description Language for SOA	[40]
Continuous refactoring' of an SOA	[24]
Service integration	[12]
Solution to defining business process	
SCA model	[8]
Analysis of enterprise business	[8]
Assembling business processes	[8, 10]
Targeting the stakeholder goals (TROPOS and MAP)	[19]
Solution to defining components	
Automatic code generation making domain models instantly executable*	[38]
Identification of the granularity of business components	[8]
Assembling business components	[8, 10]
Design (RUP)	[30]
Creation Services (XP)	[30]
Composition Services (XP)	[30]
Interface Prototype Driven Development (IPDD)*	[31]
Construction and Testing (CT)	
Solution do development	
Framework to design, implementation and deployment of SOA*	[1]
Automatic code generation making domain models instantly executable*	[38]
Development of component operations	[8, 10]
Development (RUP)	[30]
Maintenance (RUP)	[30]
Interface Prototype Driven Development (IPDD)*	[31]
Guidelines and best practices for SOA agile development	[7]
Managing change in software development (XP)	[19]
Specifying the software development process (RUP)	[19]
Agile model-driven development	[11]
Solution to governance	
E-governance system	[32]
Agile Methods governance	[9]
Solution to security	
Security framework for SOA with Agile development	[15]
Deployment and Provisioning (DP)	
Deployment	[10]
Framework to design, implementation and deployment of SOA*	[1]
Execution and Monitoring (EM)	

*The solution appear in more than one phase

Paper (CP) and Journal Paper (JP). This type of paper represents the latest in terms of publications in Computer Science. According to Montesi and Owen [27] "conference publication can measure the ability to innovate and propose new ideas, whereas journal publication can more strongly contribute to building a knowledge basis".

The studies are distributed between the years 2003 and 2017, and the peak is between the years 2010 and 2013, 9 studies. Between the years 2014 and 2016, only 3 studies and in the year 2017, 2 studies. This distribution of publication trend is following the Gartner Hype-Cycle[4]. It seems that today the subject is moving from the 'Slope of Enlightenment' to the 'Plateau of Productivity'. It is a sign of maturity of the subject. No study from the year 2018 appears in this review. This may state that the interest of the subject is decreasing or the data sources did not index the studies for the year 2018 when the data were extracted. In the future, in a possible repetition of this review, this doubt can be clarified.

5 Discussion of Research Questions

The previous section presented a general analysis of the data distribution. It was assessed what the data could explain without touching on the subject of the review itself. This section continues to analyze the data, now addressing the specific research questions.

5.1 RQ01: What Agile Solutions Are Proposed for Developing Service-Oriented Architecture (SOA) Based Applications?

Table 5 shows the type of results and the phase of the meta-model in which they were classified by the reciprocal translation process. Although all studies have addressed the RQ01, some authors have explored SOA and Agile to address issues related to business objectives and rapid changes in requirements, in the environment, and even the customer, as an alternative to traditional monolithic systems [8,9,11,12,31]. But, other authors only addressed the need to deal with the SOA and Agile itself. They assumed that the problem itself would be the combination of these two concepts. They did not explain the need to use them. [7,15,19,20,24,30,34,39].

Although some studies have presented artifacts or experiments, none of them presented results or data capable of allowing any type of validation for the readers. The general conclusion of the authors is that the proposed solutions are able to solve the initial problems presented by them. In Rong *et al.* [31], for example, the authors presented a Service-Oriented Framework of Interface Prototype Driven Development. They did a comparison between two similar projects developed by the same team. Project 'A' used the proposed framework, meanwhile the project 'B' did not. They concluded the project that used the proposed framework supports an instant response to requirements changes and improves

[4] https://en.wikipedia.org/wiki/Hype_cycle.

the quality and efficiency of analysis and design for data-centered application system, but, no data was presented that allow the readers to evaluate the results stated by the author.

Wang *et al.* [39] presented the 'SOA based Model-driven Rapid Development Architecture - SMRDA'. The authors presented a combination of SOA, Modeldriven Architecture (MDA), proposed by Object Management Group (OMG) and agile methods. They applied the SMRDA in a software education management platform. They concluded that the combination of SOA and MDA approaches is the main trend of modem software development in enterprise applications. The key of which, is modeling services correctly, and applying agile development technique.

Karam *et al.* [20] presented a tool called 'VisualWebC' (short for Visual Web Composition) to support the visual composition of Web services. The authors concluded that their approach presents several significant advantages, such as the creation of web applications with complex functionality with relatively little effort and time.

Chehili *et al.* [9] presented a platform called FraSCAti to be used with their proposed framework, the FASOAD, a Framework for Agile Service-Oriented Architectures Development. This FraSCAti platform was presented by the authors int an previous study. The authors also did a case study in a High Educational domains.

In Christou *et al.* [11] the authors presented an example using the Agile Unified process (AUP) in a process of a bank.

Abdelouhab *et al.* [1] concluded that a major benefit of the Agile-UCD-SOA (Agile & user centric SOA) framework is that it leads to highly flexible and agile software that should be able to meet rapidly changing business needs.

Chehili *et al.* [8] concluded that their approach, with successive division of the project, the customer involvement, and the acceptance test parts of the architecture can meet the terms of the manifesto of agile methods.

Shahrbanoo [34] concluded that their approach is not only practical but also viable and valuable to develop agile architecture in an agile way since it is an easy approach to apply and also emphasizes on customer involvement. All the authors concluded by the success of their proposed solutions, but there is no way to evaluate the results.

Although Chehili *et al.* [10] did not present a way to validate their results, they presented a mature work, perhaps because it is a more recent work. ASOS-DeM (Agile and Service Oriented Software Development Method) aims to overcome the complexity of web-based development projects by dividing it into subprojects to allow the application of agile methods' practices. Defines how an SOA agile development project should be executed by a self-organizing team. Describes the concepts that may be used in an SOA project such as 'Artifacts', 'Tasks' and 'Roles'. Address development, analysis, architecture elaboration, granularity identification, components assembling, deployment and integration tests, and business processes assembling.

5.2 RQ02: What Phases of the Agile Life Cycle Are the Studies Covering?

Many of the solutions are linked to the general concepts of agile methods. Some have borrowed specific methods used in methodologies that claim to be agile, such as Extreme Programming (XP), Lean Software Development (LSD), and Rational Unified Process (RUP).

Although some studies have adopted well-known development process tactics, few studies have addressed specific phases of agile development. Abdelouhab *et al.* [1] addressed design, implementation and deployment phases. Rao *et al.* [30] addressed exploration, planning, iterations to release, productionizing, maintenance, and death. Carvalho and Azevedo [7] construction phase. Christou *et al.* [11] inception, elaboration, construction, and transition phases. These results seems to corroborate the idea that defining the life-cycle in agile development methods is more difficult than in traditional development processes. Being agile is respecting other things besides processes and tools itself.

5.3 RQ03: What Common Principles of Service-Oriented Architecture (SOA) Are the Studies Covering?

Like agile phases, few studies addressed specifics SOA principles. Zúñiga-Prieto *et al.* [40] addressed service loose coupling and service composability. Roy and Debnath [32] addressed standardized service contract, service loose coupling, and service abstraction. Demchak *et al.* [12] addressed service loose coupling and service composability. The only one study that addressed all 8 SOA principles was Carvalho and Azevedo [7], standardized service contract, service loose coupling, service abstraction, service reusability, service autonomy, service statelessnes, service discoverability, and service Composability.

Service-Oriented Architecture (SOA) presents a high level of complexity and what had been observed is that few authors have entered into SOA's core. Maybe they should have put more emphasis on the details, more attention to SOA principles.

6 Conclusion

All studies in this review have presented at least one type of solution to address the development of Service-Oriented Architecture (SOA) based application with Agile methods. These studies have shown that this subject is important, worthy of study and is entering a period of maturity. However, there is a gap between research and practice. None of the studies presented a substantial result with data that can be analyzed. Few studies have attempted to present a well-defined process, although this seems to be a feature of agile methods, SOA needs some rigor in its development because of its complexity. However, this does not disqualify the combined SOA and agile methods solution. More studies need to be done, but it seems to be a promising way. This review is yet another piece to help along this way.

References

1. Abdelouhab, K.A., Idoughi, D., Kolski, C.: Agile & user centric SOA based service design framework applied in disaster management. In: 2014 1st International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), pp. 1–8 (2014). https://doi.org/10.1109/ICT-DM.2014.6917792
2. Alliance, A.: Agile manifesto, vol. 6, no. 1 (2001). http://www.agilemanifesto.org
3. Arsanjani, A., Holley, K.: The service integration maturity model: achieving flexibility in the transformation to SOA. In: 2006 IEEE International Conference on Services Computing (SCC 2006), p. 515 (2006). https://doi.org/10.1109/SCC.2006.104. https://ieeexplore-ieee-org.ez54.periodicos.capes.gov.br/document/4026979/
4. Arsanjani, A., et al.: The SOA manifesto. SOA Manifesto, p. 35, October 2009. http://serviceorientation.com/soamanifesto/original
5. Bass, L.: Software Architecture in Practice. Pearson Education India (2007)
6. Bianco, P., Kotermanski, R., Merson, P.F.: Evaluating a Service-Oriented Architecture. Research Showcase @ CMU 1 September 2007. http://repository.cmu.edu/sei
7. Carvalho, F., Azevedo, L.G.: Service agile development using XP. In: Proceedings of 2013 IEEE 7th International Symposium on Service-Oriented System Engineering, SOSE 2013, pp. 254–259 (2013). https://doi.org/10.1109/SOSE.2013.25
8. Chehili, H., Boufaida, M., Seinturier, L.: An agile approach for service-oriented architectures. In: ICSOFT, pp. 468–471 (2012)
9. Chehili, H., Seinturier, L., Boufaida, M.: FASOAD: a framework for agile service-oriented architectures development. In: Proceedings of International Workshop on Database and Expert Systems Applications, DEXA, pp. 222–226 (2013). https://doi.org/10.1109/DEXA.2013.28
10. Chehili, H., Seinturier, L., Boufaida, M.: An evolutive component-based method for agile development of service oriented architectures. Int. J. Inf. Syst. Serv. Sect. (IJISSS) **9**(3), 37–57 (2017)
11. Christou, I., Ponis, S., Palaiologou, E.: Using the agile unified process in banking. IEEE Softw. **27**(3), 72–79 (2010). https://doi.org/10.1109/MS.2009.156
12. Demchak, B., Farcas, C., Farcas, E., Krüger, I.H.: The treasure map for Rich Services. In: 2007 IEEE International Conference on Information Reuse and Integration, IEEE IRI-2007, pp. 400–405 (2007). https://doi.org/10.1109/IRI.2007.4296653
13. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9), 833–859 (2008). https://doi.org/10.1016/j.infsof.2008.01.006. http://www.sciencedirect.com/science/article/pii/S0950584908000256
14. Erl, T.: SOA: Principles of Service Design, vol. 1. Prentice Hall, Upper Saddle River (2008)
15. Farroha, D., Farroha, B.: Developing corporate services in an agile environment. In: Proceedings of IEEE Military Communications Conference MILCOM, pp. 1535–1540 (2011). https://doi.org/10.1109/MILCOM.2011.6127525
16. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer **34**(9), 120–127 (2001). https://doi.org/10.1109/2.947100
17. Idoughi, D., Kerkar, M., Kolski, C.: Towards new web services based supervisory systems in complex industrial organizations: basic principles and case study. Comput. Ind. **61**(3), 235–249 (2010)

18. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51**, 915–929 (2015). https://doi.org/10.1016/j.chb.2014.10.046. http://www.sciencedirect.com/science/article/pii/S074756321400569X
19. Ivanyukovich, A., Gangadharan, G.R., D'Andrea, V., Marchese, M.: Towards a service-oriented development methodology. Trans. SDPS **1**, 10 (2003)
20. Karam, M., Dascalu, S., Safa, H., Santina, R., Koteich, Z.: A product-line architecture for web service-based visual composition of web applications. J. Syst. Softw. **81**(6), 855–867 (2008). https://doi.org/10.1016/j.jss.2007.10.031
21. Kitchenham, B.: Procedures for performing systematic reviews. Keele University 33(TR/SE-0401), Keele, UK, 28 (2004). http://www.scm.keele.ac.uk/ease/sreview.doc
22. Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering - a systematic literature review. Inf. Softw. Technol. **51**(1), 7–15 (2009). https://doi.org/10.1016/j.infsof.2008.09.009
23. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Engineering **2**, 1051 (2007). https://doi.org/10.1145/1134285.1134500
24. Krogdahl, P., Luef, G., Steindl, C.: Service-oriented agility: an initial analysis for the use of agile methods for SOA development. In: 2005 IEEE International Conference on Services Computing (SCC 2005) Vol-1, vol. 2, pp. 93–100 (2005). https://doi.org/10.1109/SCC.2005.86
25. Lane, S., Richardson, I.: Process models for service-based applications: a systematic literature review. Inf. Softw. Technol. **53**(5), 424–439 (2011). https://doi.org/10.1016/j.infsof.2010.12.005
26. Meijer, E., Bierman, G.: A co-relational model of data for large shared data banks. Commun. ACM **54**(4), 49 (2011). https://doi.org/10.1145/1924421.1924436
27. Montesi, M., Owen, J.M.: From conference to journal publication: how conference papers in software engineering are extended for publication in journals. J. Am. Soc. Inf. Sci. Technol. **59**(5), 816–829 (2008)
28. Noblit, G.W., Hare, R.D., Hare, R.D.: Meta-Ethnography: Synthesizing Qualitative Studies, vol. 11. Sage, Thousand Oaks (1988)
29. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: an update. Inf. Softw. Technol. **64**, 1–18 (2015). https://doi.org/10.1016/j.infsof.2015.03.007
30. Rao, G.S., Krishna, C.V.P., Rao, K.R.: Rational unified process for service oriented application in extreme programming. In: 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), pp. 1–6 (2013). https://doi.org/10.1109/ICCCNT.2013.6726586. http://ieeexplore.ieee.org/document/6726586/
31. Rong, H., Zhou, N., Jin, M., Wu, J.: Research on service-oriented framework of interface prototype driven development. In: 2008 International Conference on Computer Science and Software Engineering, vol. 2, pp. 552–557 (2008). https://doi.org/10.1109/CSSE.2008.362
32. Roy, S., Debnath, M.K.: Designing SOA based e-governance system using eXtreme Programming methodology for developing countries. In: 2010 2nd International Conference on Software Technology and Engineering, vol. 2, pp. 2–277 (2010). https://doi.org/10.1109/ICSTE.2010.5608805. https://ieeexplore.ieee.org/abstract/document/5608805/

33. Severance, C.: Roy T. fielding: understanding the REST style. Computer **48**(6), 7–9 (2015)
34. Shahrbanoo, M.: An approach for agile SOA development using agile principals. Int. J. Comput. Sci. Inf. Technol. **4**(1), 237–244 (2012). https://doi.org/10.5121/ijcsit.2012.4118
35. Shi, S.S.B., Stokes, E., Byrne, D., Corn, C.F., Bachmann, D., Jones, T.: An enterprise directory solution with DB2. IBM Syst. J. **39**(2), 360–383 (2000). https://doi.org/10.1147/sj.392.0360. http://ieeexplore.ieee.org/ielx5/5288519/5386991/05387000.pdf?tp=&arnumber=5387000&isnumber=5386991
36. Timperi, O.: An Overview of Quality Assurance Practices in Agile Methodologies. Soberit. Hut.Fi 650 (2004)
37. Vallon, R., da Silva Estácio, B.J., Prikladnicki, R., Grechenig, T.: Systematic literature review on agile practices in global software development. Inf. Softw. Technol. **96**, 161–180 (2018). https://doi.org/10.1016/j.infsof.2017.12.004. http://www.sciencedirect.com/science/article/pii/S0950584917302975
38. Wang, B., Rosenberg, D., Boehm, B.W.: Rapid realization of executable domain models via automatic code generation. In: 2017 IEEE 28th Annual Software Technology Conference (STC), pp. 1–6 (2017). https://doi.org/10.1109/STC.2017.8234464
39. Wang, B., Wen, C., Sheng, J.: A SOA based model driven rapid development architecture - SMRDA. In: 2010 2nd International Conference on Education Technology and Computer, ICETC 2010, vol. 1, pp. 421–425 (2010). https://doi.org/10.1109/ICETC.2010.5529218
40. Zúñiga-Prieto, M., Insfran, E., Abrahão, S.: Architecture description language for incremental integration of cloud services architectures. In: Proceedings of 2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments, MESOCA 2016, pp. 16–23 (2016). https://doi.org/10.1109/MESOCA.2016.10
41. Zur Muehlen, M., Nickerson, J.V., Swenson, K.D.: Developing web services choreography standards - the case of REST vs. SOAP. Decis. Support Syst. **40**(1 SPEC. ISS.), 9–29 (2005). https://doi.org/10.1016/j.dss.2004.04.008