





COMPARATIVE ANALYSIS OF SOAP AND REST APIs: SYSTEMATIC REVIEW AND PERFORMANCE EVALUATION WITH PYTHON

Enes BAJRAMI ^{1*}, Agon MEMETI ², Florim IDRIZI ², Ermira MEMETI ^{1,2}

^{1*}Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University

²Faculty of Mathematical and Natural Science, University of Tetova

*Corresponding author e-mail: enes.bajrami@students.finki.ukim.mk

Abstract

In this paper, further information regarding two of the most popular web service architectures, which are Representational State Transfer (REST) and Simple Object Access Protocol (SOAP), is presented. For this, we adopted the PRISMA strategy which helped in identifying studies relevant to the chosen questions. In order to conduct the analysis of REST and SOAP, we formulated three such questions. After applying the other inclusion criteria and completion of scoring with the use of PRISMA flow chart, twelve studies were identified which met the inclusion criteria. In the process of analysis and interpretation of the selected literature, we examined many aspects of REST and SOAP relative to their performance, scalability and interoperability. Therefore, it should be understood that the goal of this particular work will be to help further this particular discussion on how likely such web service technologies will prove to be in practice. We also performed a local experiment comparing the SOAP and REST APIs in terms of performance, and thus providing additional experimental data about these technologies concerning response time and system behavior under controlled conditions.

Keywords: SOAP, REST, Web Service, Python, API

1. Introduction

Web services can be regarded as a number of applications which in unison carry out a number of operations yet are not related to one another. In this regard, there are online software application which themselves give a description, which are easy to get and easy to use with no effort [1] [2] [3]. It presents a common way of bringing together different software systems found on the world wide web and the communication between them [4] [5] [6]. In layman's terms, it can be said that web service is just aggregation of different software components that accept remote requests and return some data or services. These components are applied in developing service-oriented architecture [7] [8] [9] [10] [11] [12] approaches that transforms the internet from just a store of information to a useful resource where activities can be carried out and services accessed [13] [14]. The technology of web services also provides a cheaper and easier way for Software Engineers to efficiently develop and deploy web applications quickly. Software Engineers are able to combine their developed applications with peripheral web service components to accomplish new business processes without much difficulty [15]. Deploying web services has two recognized methodological approaches. The first is SOAP web services and the second is the REST web services [16]. SOAP is the first web service protocol which was used to send data in XML format over different protocols such as HTTP, SMTP and FTP [17]. REST is however a much simpler service-oriented architecture which works over HTTP and supports the transmission of various document formats such as XML, HTML, JSON and plain text [18] [19]. It does not matter the platform a web application is built upon or hosted; every web application is capable of utilizing web services. This time it is because of the fact that it is presented in a manner which can be understood by the computers as a web service

interface consequently [20]. This interface provides access to the functional capabilities of the web service by integrating how other programs are able to use its features. Creating web services resembles creating other computer systems, however, there are instances when bad designs result into developing web services that are not usable [21]. This can make it hard to understand what the service does and how to use it, leading to complicated and low-quality interfaces. As a result, the web service might not get used much and could be abandoned [22].

2. The State of Art

SOAP and REST are two well-known techniques that are employed in the creation of web services. SOAP (Simple Object Access Protocol) is meant to create a standard format for protocols based on XML messages, whereas REST (Representational State Transfer) depends more on non-session-based communication over the Internet which broadly employs the use of either JSON or XML to pass data across. Since SOAP presents a rigid characterized interface structure with strong typing, it is used for a higher level of integration for enterprises. Nevertheless, it is deemed not as fast since the size of the messages may be too large [23]. On the contrary, REST, makes use of sever/client architecture that is light and flexible doing away with any rigidity that is deemed uneconomical for mobile applications and web applications. Each of the two technologies wouldn't be complete without a specification for security because they provide online resources. Thus, that is what we can say in return, SOAP supports a range of features which include the ability to encrypt and authenticate messages, while REST mostly uses HTTPS and other third-part communication protocols like OAuth [24]. Both take advantage of the available tools with SOAP being the preferred choice for integration in the enterprise and REST for web API and microservice development. The factors affecting the selection process include performance optimization needs, current technology available, and how the code developers' think [25].

Table 1: Comparison of SOAP and REST based Web Services

Feature	SOAP	REST
Technology Type	Well-established traditional technology	Newer technology compared to SOAP
Enterprise Usage	Attractive in enterprises and B2B scenarios	Enterprise-ready, with successful implementations
Client-Server Interaction	Tightly coupled	Loosely coupled
Implementation	Established development kits	Interface flexibility
Changing Services	Complicated code changes on client side	No change in client-side code required
Payload	Heavier payloads compared to REST	Lightweight for efficient data transfer
Binary Attachments	Requires parsing	Supports all data types directly
Wireless Infrastructure	Not very friendly	Wireless infrastructure-friendly
Returned Data	Primarily XML data	Flexibility in the type of data returned
Bandwidth Consumption	Consumes more bandwidth	Consumes less bandwidth
Request Type	Uses POST and complex XML	Consumed using simple GET requests
HTTP-based APIs	Custom response schemas per object	Standardized URIs and HTTP verbs

Language & Platform Agnostic	Yes	Yes
Distributed Computing Environment	Designed for distributed environments	Assumes point-to-point communication
Development Complexity	Harder, requires specific tools	Simpler development
Security	Ensured through WS-Security	Relies on HTTP(S) for security mechanisms
Standards & Tooling Support	Prevailing standard with better support	Lack of support for certain standards

The first of the two positions in table 1 against the content says SOAP components against some of the components, which are of the REST Professionally Recommended. SOAP: Simple Object Access Protocol is a widely used protocol in the industry, however; REST: Representational State Transfer is just a modified version of the word SOAP. This comparison lays out areas of consideration with regards to these protocols. As regards usage within the enterprise, it is still most favorable for a lot of enterprises, particularly B2B settings with the development kits already in place and so many executed stuck up with tight client-server interactivity. In contrast, the concept of REST is already accepted as enterprise appropriate with practical applications in many respects, and allowing for flexibility in the design and looser constraints on the links. A major differentiating factor regards to the difficulty of implementation for development, SOAP applications would require heavy customization in most cases at the client end in code, while REST through its design eases changes in development removing the need to change any code. In stable messages, however, REST is somewhat heavier than SOAP transmit. In addition, there are also variations in the manner in which the data obtained from the server is processed after a query. SOAP gives back tickets in XML and REST gets back a typical data representation representing primarily JSON with a variation to what media content is delivered to a client. Ripple is an example of a SOAP based platform which swallows a huge bandwidth whose payload usage is more than that of REST which is rather less. Though Bandwidth usage for SOAP and REST also paints a picture of contrast. More so, the type and methods used in realization of the SOAP and REST will greatly affect the security usage. SOAP obliterates outer reach of security using the WS-Security standards and REST employs several applications such as HHTP(S) secure measures against threats. In spite of the web service approaches in deploying SOAP, Rest means some standards are not in the service of Rest that makes the service more complicated.

3. Related Work

Since there exists quite an extensive amount of literature on the two interfaces, it is safe to observe, the primary objective is to go back and do a literature review and try to uncover the main ideas and findings. In paper [26] the authors report on scoping review of web service antipatterns found in web service interface design. This analysis revealed the existence of twenty-three anti-patterns in the SOAP web service technologies, while only twelve were evident in restful technologies with very few of them common to the two. This is very beneficial to the developers of web services so that those antipatterns can be avoided and therefore reusable web services of superior quality are made. Such future studies are also intended to fight against the exposed antipatterns in the next works. In paper [27] the authors perform several ranking metrics on them and detailed the service-oriented architectures. They conduct a systematic literature review and use a conceptual comparison against the SOAP and REST models. Nevertheless, such conclusions do not lead to a definitive answer as to which strategy is more efficient in system integration. In this regard, the authors believe that which variant of SOAP or REST has to be implemented is to be determined in view of a comprehensive analysis including and extending to both functional and non-functional properties. For instance, they

assert that REST is better suited for the integration of lower-end information systems but not so vice versa for SOAP which is said to be more applicable in advanced systems with additional factors such as the security policies of WS-Security and other related standards. In the article, referring to systems [28] authors presented the systematic literature review that they intended to conduct when investigating its need to present the results of the research held on web services security. The SCT found out that although secure coding practices could be among the best practices applied in practice, this is not quite the case in practice, and particularly for REST-based web services. It is believed that web services should allow for non-complex data migration from one application to another application or service. Previously it made sense to say SOAP was used in an abstract manner. However, where there is the mention of REST today, there is the great elimination of elaborating which all peripheral theory is a spinning of shackles to unwritten rules and Enlightenment. Recent studies have drawn attention to the fact that these fears within RESTful web services are justified. Now, with the advancement in web service technology, the compromising approaches towards web services are also advancing. This renders studies on the security of RESTful web services crucial to prevent such forms of the attack. The cryptography that is often implemented during the development stage can also safeguard the system from data exploitation. Certainly, the path is evident, the only hurdle remaining is how to combine the most optimum means of efficient authentication and one or several encryption techniques to facilitate smooth and secure exchanging of information. In this paper [29] the authors performed a survey on the research done on RESTful services close to SOAP services and later on proceeded to the research of the services. Such papers have tend to be categorized into three main sub-groups: interaction of SOAP and RESTful architecture in different aspects, representation and realization only within REST. In terms of the first ebook and subsequent papers, they attempted to deal with these problems of consolidation of various research domains aiming at service integration including attempting to understand some papers in the first group dealing with SOAP – REST provisioning. However, they did not consider whether these types these methods were coming into conflict with the properties of REST methodology. In fact, their systematic review became problematic in the use of the first group methods in which REST and SOAP were combined with no clear rationale, a choice evidently made by the authors. In paper [30] relatively recent literature has been examined for a number of stereotypes contained in software, although starting approaches have been also reviewed. As take their identification strategies are sometimes regarded as bad smells in software as these are just some. This paper's purpose is to mainly relate to the problem specification and mostly patterns' detection in contemporary software. Instead, it does address quite a few, but not the issues connected with a specific subgeneration of modern software – RESTful software. It addresses both primitive and advanced software. The subject of the study includes the following questions, how the patterns are recognized, how they have been transformed, which patterns are harmful for some class of systems, how relation of different systems is with these patterns, and what work has been reported in the direction. These unions are helpful for our paper too as they are solutions of employing most of the recent software technology including SOAP, REST and SCA. It describes six effective approaches for figuring out bad software patterns which can be applied more or less in the current software systems. Further, it highlights five elements that have propelled these approaches. When it comes to modern applications like SOAP or REST, focus can only be made regarding bad patterns to certain extent because of the need to protect the secrets of such systems- their functionalities. It has been found out, however, that it has not been possible to recognize many low-grade patterns in the contemporary released software with respect to its software primitive; the investigation had revealed only 19. Most people seem to be skeptical on some levels basing their argument on poor patterns of a software meant for one application in relation to the same software meant for another application. In as much as all those patterns that should be in place were not there, the analysis in some parts did reveal some

efforts that have been made in order to trace the bad patterns, but virtually no advancement has been made in allowing things to move forward by either rectifying them or preventing them from returning.

4. Research Methodology

In the next section, we provide a summary on two articles; the first article compares the two Web Services and the second one discusses SOA solutions and the direction of SOA evolution. Research papers were searched on this subject with the date range between January 2018 and March 2024. At our paper we have used PRISMA for a guideline for reporting the protocol of a systematic review. It offers a formal framework in order to assist the reviewers to stipulate the aims, methodology and hypotheses of conducting a review in a bid to prevent unnecessary reviews from being conducted [31]. We have elaborated the research phases on the English section through PRISMA flowchart.

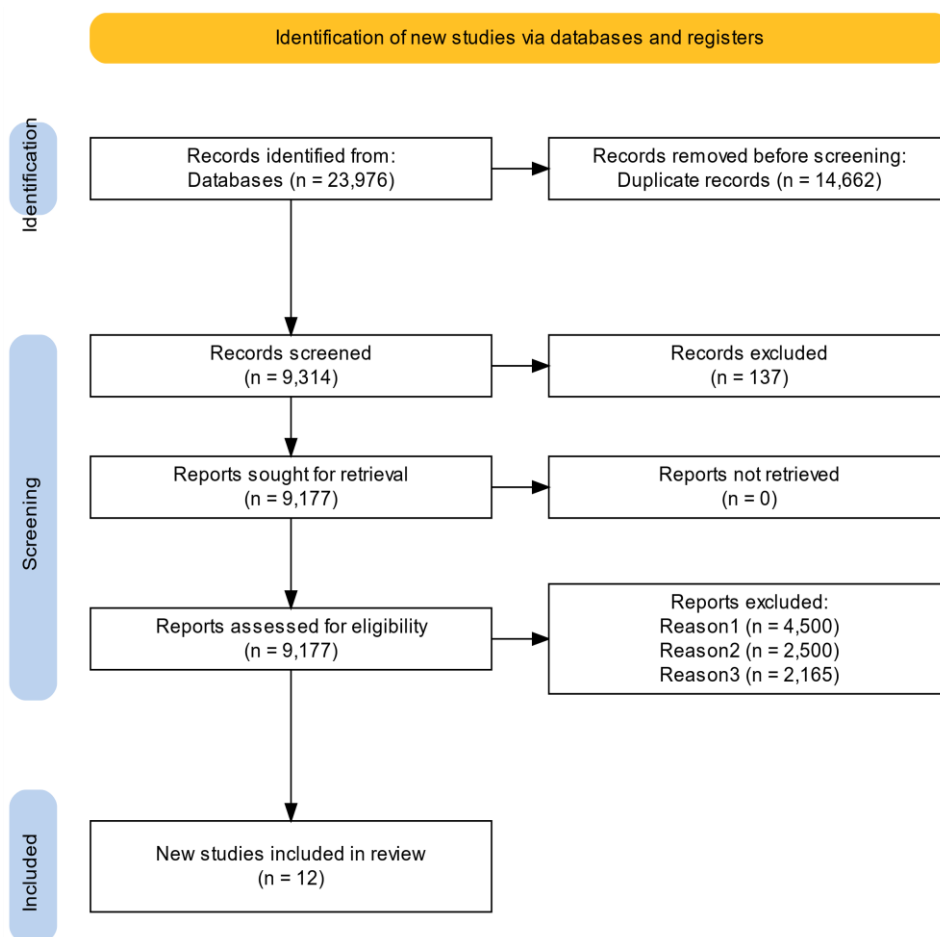


Figure 1: Flowchart of the PRISMA selection process

4.1. Research Questions: Even before beginning the search for articles on web services security, we put our heads together and thought out a few questions that would serve as our look-out. These will make it easier for us to sift through articles and find the appropriate ones in the issue. A few of the questions that we have to address in our search include:

- **RQ1:** What are the current adoption trends of SOAP and REST in modern web services, and what challenges do organizations face during their implementation?
- **RQ2:** How do SOAP and REST perform in terms of efficiency, latency, and scalability within microservices architectures under varying workloads and deployment scenarios?

- **RQ3:** What are the security mechanisms, authentication protocols, and vulnerabilities associated with SOAP and REST APIs in cloud computing environments, and how do cloud-specific factors influence their security posture?

Many papers were read in order to obtain answers to our questions, then the most important ones were selected for intense scrutiny.

4.2. Review Parameters: For our paper, we used keywords like “SOAP”, “REST”, “Web Services” to find similar stuff. Keywords are just fancy words we type into digital libraries to search for what we need.

Table 2: Keyword’s parameters

Keyword	Operator
SOAP	OR
REST	OR
Web Services	

Such keywords processed the new innovations or revise tools developed against PRISMA methodology as seen in figure 1, which was used for selection, processing and collection of articles.

4.3. Inclusion and Exclusion Criteria: The criteria of inclusion and exclusion for evaluating the relevant studies are as follows:

Inclusion Criteria:

- Only studies are written in English that were relevant to SOAP VS REST;
- Studies where experiments or similar methods were included.

Exclusion Criteria:

- Research articles not published before 2018;
- Studies that review others’ SOAP VS REST - related research work;

5. Results

A flowchart has been created in order to present the results of the search more clearly. Each of the studies was reviewed thoroughly in order to identify the most relevant information and identify the trends, with the most important data presented in Table 3. The selection strategy was demonstrated using the PRISMA flowchart, which is reflected in Figure 1, depicting our systematic approach towards the selection process in the present study. Out of discoverable literatures we found a total of 23976 literatures which were relevant to the current research area. Then there remained around 14662 separate articles after we removed duplicate articles. Then we started the screening stage of the process, and for this, in the first screening stage, 9314 of the articles were selected for the purpose of our research. However, on further inspection of the 137 articles found in the previous stage, it was discovered that they did not fit the needs of the current study and thus were left out of the next stage. The next step was to perform handpicking of the articles in the collection of articles left to obtain targeted articles that were relevant for the research resulting to about 9177 papers being obtained. Unfortunately, however, even after this careful stepwise elimination, round about 8165 papers were further not accepted according to the strict requirements. Eventually, we were able to end up with 12 review babies after the selection in process which were seen to be the most meaningful and focused for the advancement of our study. Our approach started with using a systematic method of putting specific search terms into any search engine available, to find the articles that pertain to our

subject of interest. With this careful step – we provided sufficient coverage and did not lower the level of the studies incorporated into our analysis.

Table 3: Summary of studies

#	Reference, Year	Perform Work	Contribution
1	[32], 2020	Web services at design stage of the project	The purpose of the research paper is to facilitate selection of the right web service type in the course of project design. It provides a detailed introduction to the web services, describes their structures and looks into REST and SOAP and their APIs as well. Other than that, it also looks at previous works regarding the comparison of SOAP and REST. This paper concludes by providing the recommendation of the use of RESTful web services based on the experiments conducted.
2	[33], 2019	Comparison of scalable rest application programming interfaces in different platforms	This study has developed RestAPI applications in the C#, Java, Go, Python, and Node.js languages and studied their load characteristics. Applications written in Go language were the most stable, and C# and Java were reasonably stable as well. With respect to the response, Node.js came next but comparatively lower, while in heavy load Python’s applications did poorly. These results were consistent with the principles of language design since compiled languages C# and Java proved to be better than interpreted languages, Python and Node.js. The study particularly emphasizes the importance of the language chosen in this case because of the direct compilation by Go language.
3	[34], 2018	Proposal solution for Enterprise Applications compared to Web Services	This work provides an analysis of microservices architecture vs traditional web services, with regard to the Architects’ angles of view. Employing Android mobile ecommerce smart payment application as a target case study, the two architectures were evaluated using the provided coupling huddle. The results of the application of metrics from the application showed that micro service architecture has lower service coupling compared to traditional web services.
4	[35], 2023	API Security Testing	The paper outlines peculiarities of security testing of RESTful APIs which includes advocated measures of input validation, authentication, and authorization. Insufficiently captured traceability, authorization problems, potential asset management problems and wide delayed assignment were concerns raised in the research. To complete the paper, findings are drawn and strategies on how to enhance RESTful API security with the existing research are given.

5	[36], 2021	Fuzz testing for APIs	The authors employ a black box fuzzing approach for REST APIs and address issues such as blind mutation by using Test Coverage Level feedback to determine the effectiveness of the testing methods. They improve the strategies of mutation to reduce the complexity of the testing and generate more appropriate test cases that will cover various execution paths within the REST APIs. Testing two large open-source projects reports 89 bugs but testing the APIs.guru service excavates 351 bugs in 64 hosted API services.
6	[37], 2022	Web Services Validation	The paper investigates web services in IoT, highlighting their importance in client-server interaction and inter-app communications of IoT applications. It analyzes REST API and SOAP and notes that SOAP is a messaging protocol while REST API is architecture based. This paper combines the main advantages of both for a more efficient web service deployment system and tests links in Postman. XML is primarily used for the representation of the results, with experimental data demonstrating enhanced results against earlier work.
7	[38], 2021	Development of a prototype for a web service	The article is devoted to creation of a safe prototype web service based on SPA for automating user information accounting, which is topical in the modern web services industry. It addresses the issue of web application types, architectural style choice, secure API creation process, and Python Flask framework selection. Instructions for setting up the system, configuring the database, and securing web servers are provided which helps in the appreciation of the development of a secure web service.
8	[39], 2019	Proposal Web Services to Overcome Interoperability	This research uses SOAP web services to allow users to access data in real time in the fingerprint attendance management system thus solving the interoperability problem of the devices and systems.
9	[40], 2024	Integrating blockchain with Web APIs and SOA enhances system interoperability	This research looks at how interoperability is achieved through the use of blockchain, Web APIs and Web Services within SOA, which is crucial for the linkage of different architectures. Interoperability, which has emerged as one of the most important requirements in the modern world, supports the flow of data between various systems with no restrictions. The characteristic of SOA is that it also encourages integration in structured way with support of opening standards, whereas blockchain is more of a decentralized system with

			oracles. The study compares these two approaches and analyzes the issues like trust, scalability, data integrity, transparency, governance, and regulation. Such an analysis can help organizations select the most optimal approach to achieve the interconnectivity of systems.
10	[41], 2018	Quality of service of Web services	This research investigates the evaluation of the Quality of Service (QoS) based on the users of the Web services in various settings. In the future, work will extend with new QoS parameters like failure probabilities or time of response. Also, evaluation has been done using skewness, t-test, f-test, z-test, chi-square test. The chapter determines that web services issues are application and user specific depending on the type of application and nature of the user.
11	[42], 2024	Methodology for applying attack taxonomy to web services	In this research, the objectives portray the particular process where attacks are classified and re-classified in the mid of a set of rapidly developing new attack techniques on web services. Still another set of attack classifications consisting of no less than 33 types of attacks into 5 groups: brute force, spoofing, flooding, denial of service and injection attacks, were collected in order to evaluate the present situation regarding the safety of web services. Moreover, such taxonomy of attacks is used for modelling via attack trees. This method helps combat future aggressions along many fronts but particularly web services.
12	[43], 2024	Logic vulnerability testing model in Cloud	The Rest logic authors introduce as a methodology to find logic bugs in the cloud-based REST APIs. It performs operations over API logic information to uncover interrelations and inconsistencies and obtain an information about a runtime stack of invoked procedures. Resource lifecycle testing and parameter inference for low coverage of test cases generation are also discussed. They propose that a test execution analysis technique is employed to investigate quite hidden logic faults missed within 2xx ¹ status codes. The practical implications of this work have been developed and tested successfully with the help of real-world open-source cloud Rest API logic verification proving its efficiency and inconsistency and logic faults discovery.

¹ The client's request has been successfully received, understood, and accepted by the server, indicating a successful interaction

6. Discussion

Web services have changed the course of the Internet significantly. This information has established that software engineers are beginning to embrace the new paradigm shift from web services to microservices. Many studies were conducted regarding the performance, efficiency, latency, and scalability of REST and SOAP protocols. In addition, they tried to search for any better type of web service for a specific atmosphere which would provide more security advantages. The focus of this paper is on a systematic review, while the PRISMA approach was used. Identified 23,976 total number of records. To simplify understanding and future perspective for the readers, a bar chart was created. The figure gives an overview of the circulation of papers published across diverse publication types including conference papers, journals, magazines, books, early access articles and standards respectively. This figure explains our literature review strategy and enables easy assessment for the readers. Another figure showing the shares of conference papers, conference papers and research journal articles.

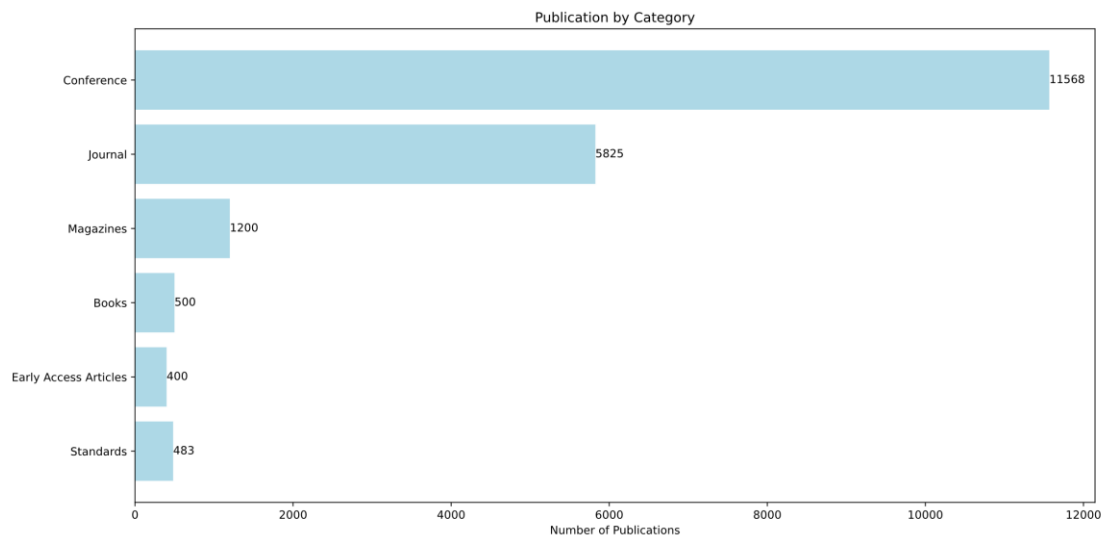


Figure 2: Publication by Category

Let's address each research question based on the contributions from the provided table:

- RQ1: What is the current scenario regarding the adoption of SOAP and REST in the current sphere of web service technology and what are the known limitations in their usage by organizations?**

Based on the examined works, it is quite apparent that today, SOAP and REST approaches are still widely present in web services landscape. Many people still prefer using SOAP over REST as ERP applications are concerned because SOAP is more robust in terms of message standards and protocol specifications. Slow communication of information probably can't be avoided but rather improved. However, organizations usually encounter various problems such as interoperability problems, security issues, performance, etc. For instance, RESTful services indeed provide better performance than SOAP based services even though the issues of security and effective data exchange are still there.

2. RQ2: Under what conditions and workloads will SOAP and REST be equally efficient among efficiency, latency, and scalability within microservices architectures?

The comparison studies analyzed in the papers shed some light on the performance characteristics of SOAP and REST in microservices architectures. SOAP is considered reliable and has defined standards whereas REST is a lighter version which often leads to better efficiency and lower latency, especially in load varying cases. Furthermore, the specific code or the utilized platform imposes particular limits on the expanding capacities of web services with respect to which compiled languages such as C# and Java are more efficient off than Python or Node.js which are interpreted languages.

3. RQ3: What are the risks, authentication tools as well as security methods of the SOAP and REST URLs within a cloud computing environment and what is the very essence of a cloud which has implications on security?

Numerous articles concern themselves with the issues of security of SOAP and REST APIs in cloud computing. They focus on issues like input checking, authentication, and authorization, and risks such as poor logging and inadequate permissions. In addition, Security Concerns also relate to multi-tenancy and shared resources in the cloud. For this purpose, new methods like fuzz testing and logic vulnerability testing are suggested that are expected to improve the security and dependability of SOAP and REST APIs in cloud computing environments.

In this analysis, the findings from the reviewed papers are assimilated in order to understand the level of adoption, functioning, and security of SOAP and REST web service protocols.

7. More experiments were done: The performance of SOAP and REST APIs is demonstrated with the use of Python

In this section, we carried out an additional experiment to compare response times of SOAP and REST- type of web services locally with the help of simple datasets in J-s-O-N format. The results indicated that the time taken for response in the SOAP API was 0.5678 seconds and the response time in the REST API was 0.1234 seconds. It is also noteworthy that the scripts were execution programmed with the help of Python, as it was noted in the text.

```
1. {
2.   "SOAP_API_Response": 1.2345,
3.   "SOAP_API_Response_Time": 0.5678,
4.   "REST_API_Response": {
5.     "data": [
6.       {"id": 1, "value": 42},
7.       {"id": 2, "value": 73},
8.       {"id": 3, "value": 15},
9.       {"id": 4, "value": 98},
10.      {"id": 5, "value": 27},
11.      {"id": 6, "value": 54},
12.      {"id": 7, "value": 81},
13.      {"id": 8, "value": 36},
14.      {"id": 9, "value": 69},
15.      {"id": 10, "value": 10}
16.    ]
17.  },
18.   "REST_API_Response_Time": 0.1234
19. }
```

Based on our results, we can draw the conclusion that even though the test was conducted in a simple setup within local environments, the REST API exhibited faster response times compared to the SOAP API.

8. Conclusion and future work

8.1. Conclusion: Lastly, it can be concluded that PRISMA methods have focused attention on comparative analysis of REST and SOAP architectures in the systematic review. Based on the synthesis of the twelve papers included in this literature review, we have reconciled the research questions and shed light on the advantages and disadvantages of both approaches. Although REST turns up to be the most common choice because of its ease of use, flexibility and being in line with current web technologies, SOAP still plays a critical role in specific areas where high security and transaction addressing are essential. The question of choosing between REST and SOAP is mostly determined by the project at hand, which emphasizes that there is a need to be careful in making architectural choices for web services. At the same time, we performed a more practical task, which was a local test of SOAP and REST APIs and their effectiveness, comparing their response times and behavior in controlled conditions.

8.2. Future Work: It is helpful to think about the future levels of these systems and how they are expected to be doing in reality and whether they will be prone to attacks in current approaches that they are quite secured. It would also be worthwhile considering how the systems can be improved to increase their interoperability, and remove use inefficiencies. For that matter, alternative architectures for developing web applications should be considered, for example, exploring web services based on microservices, serverless architectures or such where the application servers take no maintenance from the user. Perhaps, it would even be reasonable to try to understand if the use of blockchain could contribute positively to the web service technologies development. As a last point, all web services that we design in the future will integrate the understanding of all the information systems that we will create within the context of their intelligence usage. These people, as can be seen, are able to lead us to very interesting issues that I believe will also change the way web services are created and used in the future.

References

- [1] P. El-Kafrawy, E. Elabd, and H. Fathi, "A trustworthy reputation approach for web service discovery," *Procedia Computer Science*, vol. 65, pp. 572-581, 2015.
- [2] W. J. Obidallah and B. Raahemi., "A Taxonomy to Characterize Web Service Discovery Approaches, Looking at Five Perspectives," *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 458-459, 2016.
- [3] I. Lizarralde, J. M. Rodriguez, C. Mateos, and A. Zunino, "Word embeddings for improving REST services discoverability," *Latin American Computer Conference (CLEI)*, pp. 1-8, 2017.
- [4] J. Wang, P. Gao, Y. Ma, K. He, and P. C. Hung, "A web service discovery approach based on common topic groups extraction," *IEEE Access*, vol. 5, pp. 10193-10208, 2017.
- [5] M. Curiel and A. Pont, "Workload generators for webbased systems: Characteristics, current status, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, pp. 1526-1546, 2018.
- [6] M. Hirsch, A. Rodriguez, J. M. Rodriguez, C. Mateos., "Spotting and Removing WSDL Antipattern Root Causes in Code-first Web Services Using NLP Techniques: A Thorough Validation of Impact on Service Discoverability," *Computer Standards & Interfaces*, vol. 56, pp. 116-133, 2018.
- [7] M. H. I. Hamzah, F. Baharom, and H. Mohd, "An exploratory study for investigating the issues and current practices of Service-Oriented Architecture adoption," *Journal of Information and Communication Technology*, vol. 18, pp. 273-304, 2019.

- [8] M. H. I. Hamzah, F. Baharom, and H. Mohd, "A ServiceOriented Architecture Adoption Maturity Matrix using Kano Model: Cross Evaluation between IT and Business Benefits," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, pp. 105-112, 2017.
- [9] W. Dai, V. Vyatkin, J. H. Christensen, and V. N. Dubinin, "Bridging service-oriented architecture and IEC 61499 for flexibility and interoperability," *IEEE Transactions on Industrial Informatics*, vol. 11, pp. 771-781, 2015.
- [10] A. Ouni, M. Kessentini, K. Inoue, and M. Ó. Cinnéide, "Search-Based Web Service Antipatterns Detection," *IEEE Transactions on Services Computing*, vol. 10, pp. 603-617, 2017.
- [11] B. Saravana Balaji, R. S. Rajkumar, and B. F. Ibrahim, "Service profile based ontological system for selection and ranking of business process web services," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, pp. 18-22, 2019.
- [12] M. G. Galety, B. Saravana Balaji, and M. S. Saleem, "OSSR-P: Ontological service searching and ranking system for PaaS services," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, pp. 271-276, 2019.
- [13] R. B. Brinhosa, C. M. Westphall, C. B. Westphall, D. R. Dos Santos, F. Grezele, and D. R. Westphall, "A validation model of data input for web services," *Twelfth International Conference on Networks*, pp. 87-94, 2013.
- [14] J. Yu, Q. Z. Sheng, J. K. Swee, J. Han, C. Liu, and T. H Noor, "Model-driven development of adaptive web service processes with aspects and rules," *Journal of Computer and System Sciences*, vol. 18, pp. 533-552, 2015.
- [15] A. L. Lemos, F. Daniel, and B. Benatallah, "Web service composition: a survey of techniques and tools," *ACM Computing Surveys (CSUR)*, vol. 48, p. 33, 2016.
- [16] R. Gunasri and R. Kanagaraj, "Natural Language Processing and Clustering based service discovery," *International Journal of Scientific & Technology Research*, vol. 3, pp. 28-31, 2017.
- [17] A. De Renzis, M. Garriga, A. Flores, A. Cechich, C. Mateos, and A. Zunino, "A domain independent readability metric for web service descriptions," *Computer Standards & Interfaces*, vol. 50, pp. 124-141, 2017.
- [18] T. Masood, A. Nadeem, and S. Ali, "An automated approach to regression testing of web services based on WSDL operation changes," *IEEE 9th International Conference on Emerging Technologies (ICET)*, pp. 1-5, 2013.
- [19] L. Kumar and A. Sureka, "An Empirical Analysis on Web Service Anti-pattern Detection Using a Machine Learning Framework," *IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 2-11, 2018.
- [20] H. Wang, M. Kessentini, and A. Ouni, "Interactive refactoring of web service interfaces using computational search," *IEEE Transactions on Services Computing*, 2017.
- [21] J. L. O. Coscia, M. Crasso, C. Mateos, and A. Zunino, "An approach to improve code-first web services discoverability at development time," *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 638-643, 2012.
- [22] C. Mateos, A. Zunino, S. Misra, D. Anabalón, and A., "Keeping web service interface complexity low using an oo metric-based early approach," *XLII Latin American Computing Conference (CLEI)*, pp. 1-12, 2016.
- [23] Kishor Wagh and Ravindra Thool, "A Comparative Study of SOAP Vs REST Web Services Provisioning," *Journal of Information Engineering and Applications*, vol. 2, no. 5, pp. 12-17, 2012.
- [24] P. Seda, P. Masek, J. Sedova, M. Seda, J. Krejci and J. Hosek, "Efficient Architecture Design for Software as a Service in Cloud Environments," *10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 1-6, 2018.
- [25] A. Neumann, N. Laranjeiro and J. Bernardino, "An Analysis of Public REST Web Service APIs," *IEEE Transactions on Services Computing*, vol. 14, no. 4, pp. 957-970, 2021.
- [26] Fuad Alshraideh and Norliza Katuk, "SOAP and RESTful web service anti-patterns: A scoping review," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 5, pp. 1831-1841, 2019.
- [27] Juris Tihomirovs and Janis Grabis, "Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics," *Information Technology and Management Science*, vol. 19, pp. 92-97, 2016.

- [28] Eriyanto Adhi Setyawan and Fadhil Hidayat, "Web Services Security and Threats: A Systematic Literature Review," *International Conference on ICT for Smart Society (ICISS)*, pp. 1-6, 2020 .
- [29] Martin Garriga, Cristian Mateos, Andres Flores, Alejandra Cechich and Alejandro Zunino, "RESTful service composition at a glance: A survey," *Journal of Network and Computer Applications*, vol. 60, pp. 32-53, 2016.
- [30] F. Sabir, F. Palma, G. Rasool, Y.-G. Guéhéneuc, and N. Moha, "A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems," *Software: Practice and Experience*, vol. 49, no. 1, pp. 3-39, 2019.
- [31] David Moher, Larissa Shamseer, Mike Clarke, Davina Gherzi, Alessandro Liberati, Mark Petticrew, Paul Shekelle and Lesley A Stewart, "Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement," 2015.
- [32] Sarah Hussein Toman, "Review of Web Service Technologies: REST over SOAP," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 12, no. 4, pp. 18-30, 2020.
- [33] KEMER Erdem and SAMLI, Ruya, "Performance Comparison Of Scalable Rest Application Programming Interfaces In Different Platforms," *Computer Standards & Interfaces*, vol. 66, 2019.
- [34] V. Raj and S. Ravichandra, "Microservices: A perfect SOA based solution for Enterprise Applications compared to Web Services," *IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pp. 1531-1536, 2018.
- [35] Sattam J Alharbi and Tarek Moulahi, "API Security Testing: The Challenges of Security Testing for Restful APIs," *International Journal of Innovative Science and Research Technology*, vol. 8, no. 5, pp. 1485 - 1499, 2023.
- [36] Chung-Hsuan Tsai, Shi-Chun Tsai and Shih-Kun Huang, "REST API Fuzzing by Coverage Level Guided Blackbox Testing," 2021.
- [37] Jovita, J Auxily; Ramachandran, G and Edison Rathinam., "REST API and SOAP - Web Services Validation in IoT environment," vol. 20, no. 16, pp. 695-710, 2022.
- [38] D V Kornienko, S V Mishina and M O Melnikov, "The Single Page Application architecture when developing secure Web services," *5th International Scientific Conference on Information, Control, and Communication Technologies (ICCT-2021)*, vol. 2091, pp. 1-12, 2021.
- [39] Alam Rahmatulloh, Rohmat Gunawan and Irfan Darmawan, "Web Services to Overcome Interoperability in Fingerprint-based Attendance System," *Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering (IcoIESE 2018)*, pp. 277-282, 2019.
- [40] Z. A. Shaikh, K. Dahri, A. A. Memon, S. Tahseen, F. A. Abbasi and K. -U. -R. Khoubati, "Interoperability: Blockchain's Solution to SOA & Traditional Service-Based Integration Challenges," *IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*, pp. 1-5, 2024.
- [41] Maheswari S and Justus Selwyn, "A Review on the Quality of Service of Web Services," *International Journal of Mechanical Engineering and Technology (IJMET)*, vol. 9, no. 12, pp. 414-424, 2018.
- [42] Marcelo I. P. Salas, "Attack Taxonomy Methodology Applied to Web Services," *Latin-American Journal of Computing (LAJC)*, vol. 11, no. 1, pp. 68-77, 2024.
- [43] Ziqi Wang, Weihang Tian and Baojiang Cui, "RESTlogic: Detecting Logic Vulnerabilities in Cloud REST APIs," *Computers, Materials and Continua*, vol. 78, no. 2, pp. 1797-1820, 2024.

Appendix A

```
1. #Snippet 1
2.
3. C:\WINDOWS\system32>cd C:\Users\Enes\Desktop\Testing Web Services
4.
5. C:\Users\Enes\Desktop\Testing Web Services>python soap_rest_speed_test.py
6. * Serving Flask app 'soap_rest_speed_test'
7. * Debug mode: on
8. WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
   server instead.
9. * Running on http://127.0.0.1:5000
10. Press CTRL+C to quit
11. * Restarting with stat
12. * Debugger is active!
13. * Debugger PIN: 128-585-682
14.
15. 127.0.0.1 - - [15/Apr/2024 21:13:15] "GET / HTTP/1.1" 200 -
16. 127.0.0.1 - - [15/Apr/2024 21:13:15] "GET /favicon.ico HTTP/1.1" 200 -
17. 127.0.0.1 - - [15/Apr/2024 21:13:43] "GET / HTTP/1.1" 200 -
18. 127.0.0.1 - - [15/Apr/2024 21:14:12] "GET /soap-api HTTP/1.1" 200 -
19. 127.0.0.1 - - [15/Apr/2024 21:15:03] "GET /rest-api HTTP/1.1" 200 -
```

In the Snippet 1, the audience is provided with the console output when the server is being run in a local environment. The http header requests and responses logs prove that the requests were successful which can be classified by using the status code 200. One line shows request transactions made with HTTP and its respective status code.

Appendix B

```
1. #Snippet 2
2. from flask import Flask, jsonify
3. from zeep import Client
4. import requests
5. import random
6.
7. app = Flask(__name__)
8.
9.
10. SOAP_URL = "http://www.dneonline.com/calculator.asmx?WSDL"
11. REST_URL = "https://jsonplaceholder.typicode.com/posts"
12.
13.
14. @app.route('/soap-api', methods=['GET'])
15. def soap_api():
16.     try:
17.
18.         client = Client(SOAP_URL)
19.         result = client.service.Add(random.randint(1, 100), random.randint(1, 100))
20.         return jsonify({'response': result})
21.     except Exception as e:
22.         return jsonify({'error': str(e)})
23.
24. @app.route('/rest-api', methods=['GET'])
25. def rest_api():
26.     try:
27.
28.         response = requests.get(REST_URL)
29.         data = response.json()
30.         return jsonify({'response': data})
31.     except Exception as e:
32.         return jsonify({'error': str(e)})
33.
34. if __name__ == '__main__':
```

Code Snippet 2 deals with both SOAP and REST based web services using Flask as a platform. By means of Zeeb, it is easy to invoke SOAP services and requests library is used for RESTful operations. The SOAP API request is directed towards the calculator's service at 'http://www.dneonline.com/calculator.asmx?WSDL', which is able to perform addition of two random integer operands by the clients. Conversely, the REST API endpoint makes use of a GET request to fetch data from 'https://jsonplaceholder.typicode.com/posts' and responds with the JSON data obtained. This snippet gives a brief example of how you can implement SOAP and REST APIs in a single Flask application which would help in integration and communication of the APIs within the application.