Journal of Advances in Developmental Research (IJAIDR)
E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Comparative Analysis of Soap and Rest-For Enterprise-Level Applications

Hareesh Kumar Rapolu

hareeshkumar.rapolu@gmail.com

Abstract

Web services are increasingly becoming integral in enterprise environments due to their handiness and reusability. The two architecture of developing web services for enterprise level applications are Simple Object Access Protocol (SOAP) and Representational State Transfer (REST). However, users often require more than a single architectural design to satisfy their demand. Accordingly, it is critical to blend web services to achieve specific enterprise applications. However, most assessments of SOAP and REST do not address the overall aspects of the two technologies in enterprise level applications. This paper addresses this gap by from an enterprise perspective. The architecture of the two technologies is first examined followed by comparative analysis.

Keywords: **Simple Object Access Protocol (SOAP), Representational State Transfer (REST), Enterprise Applications, Web Services, Architecture**

I. INTRODUCTION

The increasing dependence on distributed systems and web services in enterprise environments demand a strategic selection of communication architecture. Today's enterprises operate in an ecosystem where integration requirements for disparate system, secure information sharing and
scalability are critical [1]. There are two dominant paradigms in web service communication: Simple Object Access Protocol (SOAP) and Representational State Transfer (REST), each with unique methodologies, benefits and applications. SOAP was the initial web service protocol used in data sharing across different protocols such as FTP, SMTP and HTTP in XML format [1]. It was first Microsoft and IBM in the late 1990s but has evolved in a more superior protocol reinforced by W3C standards [1]. SOAP leverages Web Services Description Language (WSDL) to define service contracts [2]. REST, introduced by Roy Fielding in his 2000 work, emerged as a simplified service-oriented protocol that applies on HTTP [2]. It supports the flow of data in formats such as plain text, JSON, HTML and XML [1, 2]. The building or hosting platform is not an important consideration given that all applications can utilize web services [3]. The rationale is that computers understand applications as a web service accordingly [3].

Enterprise applications often grapple with a myriad of challenges ranging from integration of legacy systems to maintaining high security levels to scalability needs to match changing demand [1,2,3]. Financial institutions, for example, SOAP may become a priority to secure transactions given its superior security protocols and assured message delivery [3]. Conversely, REST may become a better option in a social media platform where handling of millions of API calls per second requires scalability

and lightweight structure [4]. Therefore, a comparative analysis is critical in the context of increasing adoption of mobile-first strategies, hybrid cloud environments and microservices architectures for flexibility and efficiency of web-service communication.

II. SOAP AND REST ARCHITECTURE

The two dominant techniques are widely used in web services creation. While SOAP is designed for the creation of standard format of XML-based protocols, REST relies on a set of principles (simplicity, scalability and stateliness) to guide the design of web services [4, 5]. SOAP architecture follows the World Wide Web Consortium (W3C) standards and focuses on interoperability and consistency across different programming environments and platforms. SOAP messages are made of envelop (defining the start and end of the message), header (optional metadata related to the message) and body (holding the main content of the message) [5]. SOAP is used in enterprises requiring high integration levels given that it is defined by a rigid interface design. Yet, it is considered slower than REST in cases of too large message sizes [6].

Conversely, REST adopts a resource-oriented architecture. It models the web as a set of resources identified by Uniform Resource Identifier (URI) [7]. Each resource represents an entity like order, user profile and product. These resources are subject to manipulation through standard HTTP methods like GET (retrieve), POST (create a new entity), PUT (update) and DELETE (remove) [5].

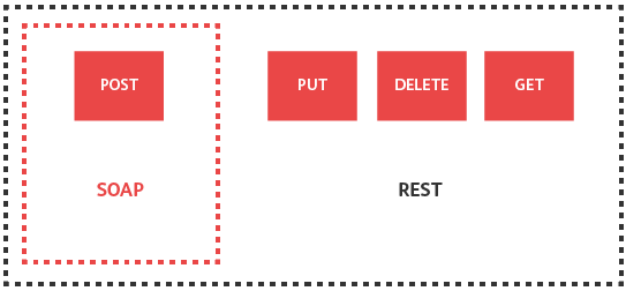

Figure 1: SOAP and REST Request Methods

The two technologies are not fully understandable without considering security as they both offer online resources. As such, it is conceivable that SOAP is characterized by a range of features that support encryption and authentication capabilities [7]. REST, on the other hand, mostly utilizes HTTPS in addition to other communication protocols [8]. SOAP and REST leverage relevant tools where REST is preferred for microservice and web API development while SOAP is used for integration of enterprise resources [8]. Accordingly, the choice of either technology is dependent on factors such as performance and payload, security, scalability, error handling and interoperability.

Table 1: Summary of Architectural Aspects

Architectural Aspect	SOAP	REST
Communication Model	Protocol-based with strict standards	Resource-oriented (HTTP methods)
Message Format	XML	JSON, XML, HTML, Plain Text
Transport Protocol	Transport-independent (HTTP, SMTP)	Primarily HTTP

Interoperability	Strong via WSDL	Flexible but lacks formal contracts

III. COMPARATIVEANALYSISOF SOAP AND REST

1. Performance and Payload

SOAP supports advanced features like attachment capability through MTOM [9]. However, its overall performance decreases when handling large message volumes or constrained network bandwidth. The heavyweight design in SOAP prevent efficient performance when handling large quantities of data generated and processed by Internet of Things (IoT) systems [10]. Conversely, the simplicity of REST gives it a clear advantage of performance. The data exchange through JSON provides RESTful services with ease of parsing than XML standards do [6]. Fast data transfer becomes possible due to JSON's lightweight structure that speeds up serialization/deserialization time, minimizing latency in high-throughput systems [6]. To illustrate, REST delivers efficient payload management in mobile applications due to minimized server data exchange costs [11]. Its approach creates improved user experience. REST and its stateless structure also minimizes server overhead to maintain server performance at scale.

2. Security

SOAP delivers powerful security capabilities which make it an appealing option. It integrates WS-Security advanced security capabilities to enable message-level encryption authentication and data integrity functions [12]. SOAP establishes secure data transmission regardless of intermediaries traversed to reach the target destination. The data requirements such as HIPAA or PCI DSS in banking and healthcare systems require SOAP due to its built-in digital signature functionality, encrypted credentials and secure message routing capabilities. The security regime of REST, conversely, depends on HTTPS because this protocol meets common web application security needs. Yet, it does not provide advanced options like SOAP for messaging. REST works well in applications with moderate security demands (for example, e-commerce platforms). Yet, there is need for measures beyond HTTPS for highly sensitive use cases.

3. Scalability

Statefulness in SOAP generates a scalability challenge for highly dynamic systems. Mounting session states utilizes significant server resources which might slow down operations when concurrent users increase. SOAP-based systems for large-scale enterprise resource planning (ERP) applications often face efficiency in scaling operations when user activity surge [5, 6]. However, the stateless design of REST makes it more flexible because each request contains all processing information. Accordingly, cloud-native applications adopting microservice architectures use REST because its scalability enables management of distributed systems and high-traffic APIs [1].

4. Interoperability

SOAP achieves higher interoperability through its dependence on WSDL for creating service contracts. SOAP delivers consistent platform communication which makes it a competitive solution for enterprise integrations of legacy systems [6]. A multinational corporation can use SOAP to achieve uninterrupted communication between different ERP platforms across its subsidiaries systems. The flexibility and widespread adoption of REST do not come without disadvantages. It generates inconsistencies in

resource representation and API design [12]. RESTful service interoperability relies heavily on documented API standards. Yet, different providers follow divergent API designs [13]. The effectiveness of REST in public APIs like social media platforms depends strongly on well-documented methods in addition to best practices [14].

Table 2: Summary of Comparative Analysis

Comparison Aspect	SOAP	REST
Communication Model	Protocol-driven with strict contracts	Resource-oriented using HTTP methods
Performance	Heavy XML-based payloads: slower processing	Lightweight JSON: faster and more efficient
Security	Advanced WS-Security	HTTPS-based: less advanced
Scalability	Stateful: limited scalability	Stateless: highly scalable
Interoperability	High with WSDL	Flexible but less formalized

IV. CONCLUSION

The synthesis of extant knowledge shows that REST is the most dominant choice based on its design around flexibility, simplicity and adherence to web technologies. However, SOAP still maintains significant role in domains that demand high security levels. The choice between the two technologies is largely subject to project specifics. The analysis suggests the need for cautious consideration when choosing SOAP or REST as the most preferred archicteratural option for enterprise web services. Equally, there is need for more practical tests on the effectiveness of the two architectures, focusing on performance metrics such as response time, error handing and overall effectiveness.

V. REFERENCES

[1] E. Y. Jesus, B. Gueye, and I. Niang, “A comparative analysis of SOAP and REST web service composition based on performance in local and remote cloud environments,” in *Proc. 4th Int. Conf. Networking, Information Systems & Security*, Apr. 2021, pp. 1–8.

[2] M. W. Khan and E. Abbasi, “Differentiating parameters for selecting simple object access protocol (SOAP) vs. representational state transfer (REST) based architecture,” *J. Adv. Comput. Netw.*, vol. 3, no. 1, pp. 63–, Mar. 2015.

[3] H. Subramanian and P. Raj, *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Birmingham, UK: Packt Publishing Ltd, Jan. 2019.

[4] U. Riaz, S. Hussain, and H. Patel, "A comparative study of REST with SOAP," in *Multimedia Technology and Enhanced Learning: Third EAI Int. Conf., ICMTEL 2021, Virtual Event, Apr. 8–9, 2021, Proc., Part I 3*, Springer, 2021, pp. 485–491.
[5] J. Sayago Heredia, E. Flores-García, and A. R. Solano, "Comparative analysis between standards oriented to web services: SOAP, REST and GRAPHQL," in *Applied Technologies: First Int. Conf., ICAT 2019, Quito, Ecuador, Dec. 3–5, 2019, Proc., Part I 1*, Springer, 2020, pp. 286–300.
[6] F. Halili and E. Ramadani, "Web services: A comparison of SOAP and REST services," *Mod. Appl. Sci.*, vol. 12, no. 3, pp. 175–, Feb. 2018.
[7] A. Soni and V. Ranga, "API features individualizing of web services: REST and SOAP," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, no. 9, pp. 664–671, Aug. 2019.
[8] C. M. Garcia and R. Abilio, "Systems integration using web services, REST and SOAP: A practical report," *Rev. Sist. Inf. FSMA*, vol. 1, no. 19, pp. 34–41, 2017.
[9] B. Simon, B. Goldschmidt, and K. Kondorosi, "A performance model for the web service protocol stacks," *IEEE Trans. Services Comput.*, vol. 8, no. 5, pp. 644–657, Jul. 2014.
[10] C. C. de Melo Silva, H. G. Ferreira, R. T. de Sousa Júnior, F. Buiati, and L. J. Villalba, "Design and evaluation of a services interface for the Internet of Things," *Wireless Pers. Commun.*, vol. 91, pp. 1711–1748, Dec. 2016.
[11] M. Melnichuk, Y. Kornienko, and O. Boytsova, "Web-service. RESTful architecture," *Автоматизациятехнологических и бизнес-процессов*, vol. 10, no. 1, pp. 17–22, 2018.
[12] K. Wagh and R. Thool, "A comparative study of SOAP vs REST web services provisioning techniques for mobile host," J. Inf. Eng. Appl., vol. 2, no. 5, pp. 12–16, Jul. 2012.
[13] S. Kumari and S. K. Rath, "Performance comparison of SOAP and REST based web services for enterprise application integration," in Proc. Int. Conf. Advances Comput., Commun. Informatics (ICACCI), Aug. 2015, pp. 1656–1660.
[14] S. H. Toman, "Review of web service technologies: REST over SOAP," J. Al-QadisiyahComput. Sci. Math., vol. 12, no. 4, pp. 18–, Nov. 2020.