ISSN:0975-9646

SOA AND REST Synergistic Approach

Navdeep Dahiya[1], Neha Parmar[2]

[1] *IBM India Pvt. Ltd, Uttar Pradesh, India*
[2] *makemytrip.com , Haryana, India*

Abstract— APIs are everywhere, reusability, resource-sharing, monetization counter for their ubiquitous nature. The true value of an API comes from lifting the business beyond the walls of an enterprise by delivering business solutions to the partners, expanding the customer base, building a community around a brand, driving more traffic towards a website, acquisition of other businesses. With APIs, business gains power with content and data ownership . The business-driven nature of APIs and the increasingly reliable Web services and cloud-based applications, have truly become one of the most crucial elements in the process of gaining businesses the momentum they never had. APIs are revolutionizing the world of IT by making it possible for developers to connect myriad applications to accelerate strategic business execution. We have been using services to define business functionality or business process were defined in terms of services. SOA had its hand in the development of such rife architectures. Providing a single point for applications to access services bolstered business a lot. With more and more technological standards being defined. It is becoming really one of foremost decisions that businesses should understand and choose to make as to which architecture they want to base their services on.
In this paper we discuss about how to build a platform for this growing ecosystem?
What should be architectural components for building such a stronger system?
What all technology components should be involved? How we should organize these components?
What are the main differences between service and resource based architectures? Is ROA future of interoperability?

Keywords— API, Services, Resource- Orientation, REST, RESTful services, SOAP and WS-*.

1. INTRODUCTION

A successful API must offer a value to those who are going to use it and to the enterprise offering it. With the concept of programmable web becoming more and more popular, new APIs are published every day. Few most coveted features of an API might be:

- Stability
- Layered Systems
- Security and Authentication
- Fast Responses
- Uniform Interfaces
- Appropriate to audience

Since public APIs are forever, design for these should be robust, scalable, evolvable and modular. SOA use to drive this effective API design [1]. SOA is more than merely a technology. It offers a strategic way to support business agility. Business concerns today in the area cover from discovery of services, definition, design, deployment and publishing of the composite and atomic services. With the advent of REST it is now thought that REST is the future of SOA. Very first assertion here would be that REST is also just an architectural style of developing resources of web as SOA does by providing services. In this paper we try to corroborate a dispassionate outlook on both styles of application development. We will discuss if both are emerging as competitors, or both are working towards a bigger unified whole.

Forthcoming sections in this paper are given to see how can we initiate the process of setting up business services using the two approaches.

In upcoming section we discuss the architectural components and their relationships to each other—as well as to core SOA application components available for APIs.

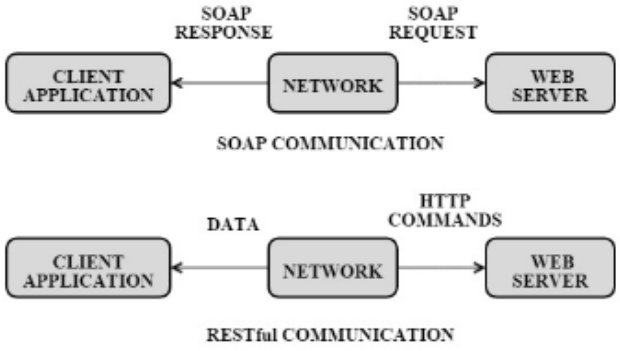

Fig. 1: SOAP AND RESTful COMMUNICATION

2. LITERATURE REVIEWED

SOA architecture mainly emphasized decomposition of business function into smaller more manageable components thus simplifying business processes. Doctrine of SOA architecture advocates [1,5, 9]:

- Defining repeatable business task/functions, that is a smallest independent task in an otherwise complicated business service.
- These functions then collaborate to solve a basic business problem which might be an aggregation of many smaller business functions, these are called services.
- A architectural style based on Service Oriented principles, consists of thoughtfully defined reusable business services address commonest of business concerns.
- This decoupled way enables externally developers to be able to use these services using their simple interfaces [11].

As opposed to this REST has two main approaches: true REST and RESTful technology approach for defining services. REST is an architectural suite that promulgates guidelines for organizing services. ROA (Resource-

Oriented architecture) is based on resources. It defines decoupled distributed components handled through a common interface.

Most important question here is, how are services and resources different in that respect.

A service is a smallest business process, independently developed, deployed, managed, and maintained software program providing business functionality for an enterprise as a whole and is open by design. It can be described like searchItem(String itemkey);. Functionality is exposed through an interface discoverable by the outsiders.

Resources on the other hand are directly accessible, deployed, managed software artifacts providing specific information or data.

With REST different operations can be defined on HTTP operations as follows [5]:

- createData - Create a new resource (and the corresponding unique identifier) - PUT
- getDataRepresentation - Retrieve the representation of the resource – GET
- deleteData - Delete the resource DELETE
- modifyData - Modify the resource – POST
- getMetaInformation - Obtain meta information for the resource - HEAD

Resources are accessed using URI.

2.1 Difference between REST and SOA

As follows from the discussion above SOA and REST are very different methodologies.

If Web services areas the RPC of the Internet, REST is the DBMS of the internet [4]. Traditional SOA based integration visualizes different software processes being able to interact with each other through methods i.e. using object oriented concepts. REST effectively allows each software artifact to behave as a set of tables, and these processes talk to each other using SELECT, INSERT, UPDATE and DELETE. (GET, PUT, POST, DELETE). Having said that ROA and SOA are very different architecture which will seldom be developed together [2,3]. However a unified approach is possible utilizing RESTful resources which we will analyse in the next section.

2.2. REST COMPATIBILITY WITH SOA

The REST Web Service approach is what is being adopted these days. The REST Web Service approach uses REST only as a communication technology to build SOA. Services will be defined using SOA style decomposition and REST-based Web Services are used as a transport. REST Web Services have nothing to do with true REST and is similar to how services communicated using XML over SOAP with only difference that it supports multiple other data types ranging from JavaScript Object Notation to binary blobs and leverages additional HTTP methods, which is commonly based on GET and PUT [3]. JSON became popular due to advancement in AJAX technology on the client side. This way rest provided a better way for providing services to heterogeneous platforms like android, java web applications, .Net Applications. Among the other differences between SOAP and REST is that while REST is implemented directly on top of the HTTP protocol, SOAP introduces heavy SOAP messages for communication to the services. Although both have their merits, trying to limit SOA implementation to a single transport – HTTP, isn't really a good idea [12].

	SOAP	REST
Protocol for invoking operations	SOAP	HTTP
Acquiring interface definations	WS-MetadataExchange	No standard
Defining policy	WS-Policy, et al.	No standard
Data formats	XML	XML,JSON, others
Conveying security tokens	WS-Security	HTTP, SSL
Establishing a security context	WS-SecureConversation	SSL
Language for describing interfaces	WSDL	No standard
Providing end-to-end reliablity	WS-ReliableMessaging	No standard
Supporting distributed ACID transactions	WS-AtomicTransaction, WS-Coordination	No standard
Transport Protocol	HTTP, TCP, others	HTTP

Fig.2: Concise comparison on the communication formats of web, reason for increasing REST usage is apparent.

With REST both requests and responses can be short. Reasons for this are that SOAP messages are wrapped by XML envelope around every request and response, thus increases the size of a message [3]. The important thing to consider here is not how much size the envelope adds, but rather the portion of overhead it creates. Since the message envelope size remains the same, impact of overhead decreases as the size of the message grows, eventually becoming negligible. For larger requests the size of the request and response is quite large and consequently the overhead of a SOAP envelope will not pose a problem[13]. Also, REST, provides a lightweight messaging alternative – JSON.

2.3. Aligning API development

Determining the architecture for a robust API Infrastructure one should devise a business strategy from SOA development to development of APIs. The strategic steps might vary from enterprise to enterprise.

The basic steps that form the backbone for any business might comprise the following:

- Identify and document your business tasks at grass-root level.
- Establish business function ownership, funding, reporting and analytics.
- Build modular services in a way that services can be reused.
- Catalogue information about services like, terms and conditions of usage, security requirements.
- Set up an integration framework for use by business partners.
- Devise policies that make your keep your business services secure and reliable.
- Monitor and analyse usage patterns, any access attacks, service performance , etc., to ensure business functions availability, relevance and returning business value.
- Manage the business functions within portfolios, separating differing business concerns and thereby guiding where to invest and how to maximize business return, as well as how to make your business more agile.[8]

2.4. Technological architecture for APIs

Service Oriented Architecture is a design pattern in which all business functions are modelled in a way to keep them independent. This loosely coupled code enables applications to be developed from these services from time to time. Developers can create new services whenever business needs just by congregating these smaller business components. Main idea of Service Oriented Architecture is that an organization can extend its services at any point of time as and when business demands since creating a new service becomes as simple as reusing existing services to accomplish a business task. The Service Oriented principles enable services to be reusable, flexible and open for enhancement. With uniform interfaces (using which services are accessed) and modular design implementation keeps both publishing/exposing of services and their internal implementation totally decoupled so that any changes to the services will not impact any external user as long as the service contract remains the same as partners will always use these services using exposed interfaces. This interoperability can be ensured by adhering to open technologies. Interoperability is challenging because Web services protocols for message exchanges, reliability and optimizing services are varied and services may be hosted on adverse platforms. Without a platform designed around standards and aimed at interoperability, it is impossible to quickly bind services together to meet continually changing business requirements or agility. Quality of service protocols optimizations will be needed as independent infrastructure as cross cutting functions that can be enabled independent of business logic and the implementation strategy for services. The key standards to support this are the Service Component Architecture and JAX-WS.[14,11].

In the next section we illuminate cardinal rules or standards and open technologies for optimizing architecture providing highly available, real-time, reliable services.

2.5. Basic Requirements of SOA

This paper is emphatic about choosing best of the different approaches available for interoperable open services. We jolt down some basic requirements here[5]:

Messaging*:* Clear and interoperable standards for establishing communication between services are the basis for interoperability. For services to communicate with each other, messages are en-coded according to the SOAP standards 1.1 and 1.2 specifications, and typically sent over HTTP. With RESTful resources as discussed above we will be able to utilize http for communication.

Service description and discovery: Services published should be available to external developers for location and discovery. Description should be adaptable to a variety of formats and platforms. Using API gateways or other available solutions are a good bet here.

Implementation: Loosely coupled code is always appreciable and extensible, keeping services ready for the future.

2. SOA Advance Infrastructure Requirements

The demands put on SOA to ensuring availability, scalability, reliability and optimization of SOA-based services and applications. Infrastructure services required to support these are[7, 9, 8,10] :

TCP Connection management: TCP Multiplexing reduces the impact of huge calls placed by SOA or WEB. This also improves performance by removing the overhead of switching involved in opening and closing multiple TCP sessions (connections) between the client and the server.

Load Balancing: High Availability of a service is an ability of the system to be able to continue to perform even when down with component failure. Not only this, services should also respond in specified amount of time. Availability can be achieved by distributing load amongst multiple data centres or by providing failover capabilities to address calls when one site is inaccessible or goes down so that these calls can be routed to some secondary site that acts as the failover capability.

Advance health monitoring services: Advance health monitoring capabilities include the ability to dynamically define routing decisions based on the message returned. Many health check services use general “ping” based services to alert when the server stop responding or if its efficiency drops by a pre-specified threshold that alarms the developers to take necessary action of rebooting or taking measures to bring loads and its performance in place.

Optimization: Optimization includes the implementation of protocol-specific standard enhancements that increase the performance of core protocols such as TCP and HTTP.

It can be performed by offloading certain resource critical operations such as SSL encryption and decryption from services to the infrastructure or through the using compression technology to reduce the total size of messages. XML-based messages are particularly well suited to compression because they are text-based. Infrastructure extensibility could be a major step in optimization but with so much boom in cloud computing these days systems can leverage from available infrastructure services on the fly as needed.

Logging/Auditing: Logging is writing the details of events happening in an application to a persistent store. Logging can be about recording normal events as well as abnormal events such as errors or exceptions. The purpose of logging is mainly from the operations perspective to get the operational metrics as well as to help ensure a service-level agreement (SLA). It also gives ways to set up monitoring and analytics for services. Persistent store where audits can be maintained might be databases or memcached as it suitable for the enterprise.

3. REST AND SOA INTEGRATION

By now we understand that REST and SOA both are architectural styles for designing networked applications targeting slightly different aspects which together are going to set the stage for ROA (REST Oriented Architecture) with more secure, simple, reliable APIs. The idea is that, rather than using convoluted conventional mechanisms such as RPC or SOAP to connect between machines, calls between machines are made through simple HTTP. This being lightweight, enhances the efficiency and evolves as an alternative to Web services[6]. Let alone protocol usage, ROA chooses best of the two areas and thus will definitely revolutionize APIs for good. We discuss below the constraints imposed below by REST and identifying their suitability in the current SOA setting [3, 2,10].

3.1. REST Principles and Constraints:

Client - Server Communication: Keeping internet protocols, exchange formats common so that diverse client platforms can easily connect to server-side resources. This can be achieved using RESTful services as they support a good range of messaging formats for communications.

Stateless Communication: No state is stored at the server, Client request contains all the information to process the requests. RESTful system mark data as cacheable. Cacheable data decreases network traffic and reduces back-end system load. For making the system the data might be cached to serve repeated traffic for similar requests. In order to utilize REST Developers should defer from storing anything shared by the client.

Uniform interfaces: In order to scale up to enormous web of interfaces and services in the internet using simple and uniform interfaces is the key to integrate with both more resources and more users. It decouples the architecture enabling each part to evolve independently. Web developer and Web API administrator both need not worry about the issue that otherwise might create a havoc.

***Layered System*:** An API Facade hides internal implementation complexity, and presents a simple interface to external consumers. Simple RESTful API interfaces hide multiple back-end databases and aggregated services. Exposing complex back-end services to Web standards like HTTP could be cumbersome job for enterprises. API Facade gives a simple pattern for publishing complex APIs. It gives buffer or virtual layer between the interface on top and the API implementation on the bottom. API endpoints are lightweight proxies enforcing the security, monitoring the usage, and shaping the traffic. The proxy facilitates a clear separation of concern between consumer interface contract and back-end service implementation.

The constraints allow beneficial properties to emerge, namely simplicity, scalability, modifiability(agility in the underlying system), reliability(Through failover mechanisms or ensuring high availability), visibility(Use of open interfaces), performance(Load balancing dynamically), and portability. RESTful APIs are created when services are to be offered outside of native application domain to a larger consumer base out of an enterprise. Like SOA, REST is an architectural discipline defined by a set of design principles, and a set of architectural constraints is also imposed by REST. A resource-centric model is used in REST; resource-centric model is the inverse of an object-centric model . In REST, every thing of interest is a resource. When modelling a RESTful service, the service's capabilities are encapsulated and exposed as a set of resources.

4. PROTOCOL SUPPORT AND USAGE

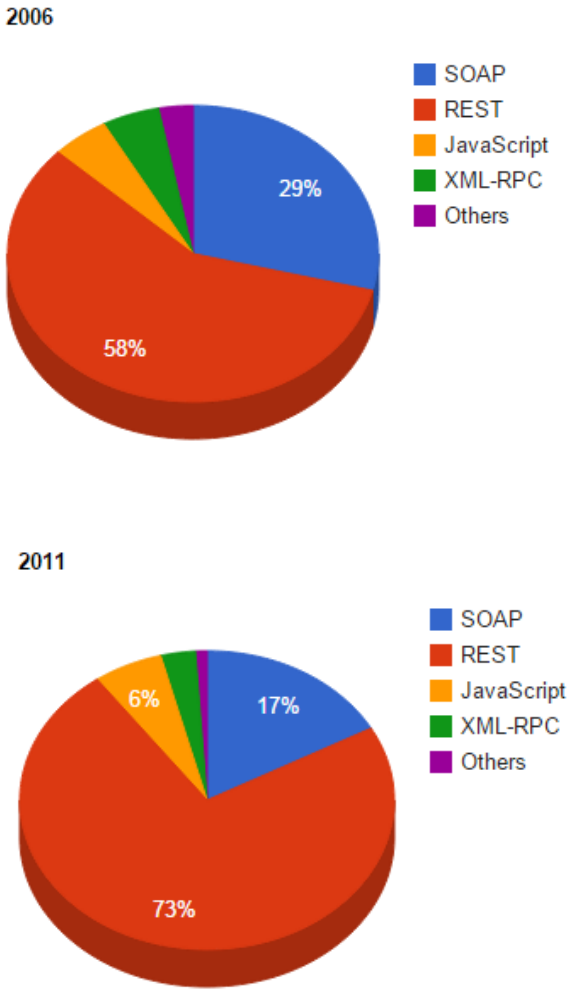

Fig. 3: Above two charts show data statistics collected by Programmable web to study patterns in protocol usage and adoption overtime, it can be seen RESTful resource as being welcomed to a good extent.

5. CONCLUSION AND FUTURE SCOPE

ROA can be thought of a strategy to align IT assets with business capabilities in a RESTful manner, thoughtful designing of strategic components utilising the best of both SOA and REST prove to be a good investment.

SOA's strong focus on sharing and re-use can optimize IT asset utilization and REST focus on extensibility and secure ways of communicating to external systems, together might prove a great tools for enterprises. Most intriguingly, ROA is supposed to re-invent business-to-business interactions, enabling even better partner relationships, and that it did quite efficiently, that will also address the main concerns in the area that is how these services be available on diverse platforms and to all the external developers to seek to use existing APIs and support the process networks. External services became a mechanism to extend an enterprise's ecosystem extension by reducing the cost of interaction, incorporating the external business capabilities, enabling the business specialization, and creating solutions of higher value that extend business processes across a partner network.

API-centric Enterprise organizations reach more customers and generate greater business revenue, build better brand value. APIs facilitate interconnecting business processes across the extended value chain. Customers, distributors, suppliers and partners can readily tap into a business capability which is offered as an API, and increase the business interaction over the API channel [6]. API-centric Enterprises move beyond destination ecommerce to embrace decentralization, personalization, contextualization, ramification, and dynamic distribution channels. Technology trends incorporating these concepts include mobile, machine-to-machine, person-toperson, and business-to-developer channels. In each case, APIs are the connecting channels across distributed solution actors and components.

Open systems pose a heavy requirement for secure transactions. REST can be employed to achieve these transaction. Provided APIs are bound to evolve over time, a clear definition should be provided as to how business are going to deprecate existing service or redirect to the one's in place. Implementing ROA remains a big challenge but businesses need to pay heed to the ways they can make their businesses more agile [13]. With this investment developing application will become really easier, with less amount of care needed by API administrators. As far as statelessness is concerned it quite obvious that some complex APIs perform some huge computations and might keep this data for future, if data needs to be saved business objects should be defined in such a way that no data is compromised. The extra burden on developers part is worth the effort because it confers agility and lowers platform choke-in and will impart improved strength to API Economy.

REFERENCES

[1] Md Tanvir Ahmad, Prof. M. Afsar Alam, Shah Imran Alam. "SOA Approaches Analysis and Integration with Emerging GUI"

[2] James Nahon. "A Comparative Analysis of REST and SOAP"

[3] Roberto Lucchi, Michel. "Resource Oriented Architecture and REST"(http://inspire.ec.europa.eu/reports/ImplementingRules/network/Resource_orientated_architecture_and_REST.pdf)

[4] (http://blog.dhananjaynene.com/2009/06/rest-is-the-dbms-of-the-internet)

[5] Lori MacVittie - "SOA Infrastructure Reference Architecture: Defining the Key Elements of a Successful SOA Infrastructure Deployment"

[6] Haddon Hill Group – "API Management & Convergence with SOA"

[7] "Accelerating API Adoption through Lifecycle Management"

[8] Mulligan, G. , Gracanin, D., IEEE "A comparison of SOAP and REST implementations of a service based interaction independence middleware framework"

[9] (http://www.oracle.com/technetwork/topics/next-generation-soa-infrastructure-132959.pdf - Next-Generation SOA Infrastructure)

[10] Brian Mulloy - "API Façade pattern"

[11] (https://www.f5.com/pdf/white-papers/soa-infrastructure-reference-wp.pdf)

[12] David Chappell – SOAP vs REST: Complements or Competitors

[13] Springer - WonSeok Lee,Cheol Min Lee,Jung Won Lee,Jin-soo Sohn - "ROA Based Web Service Provisioning Methodology for Telco and Its Implementation"

[14] Gavin Mulligan and Denis Gracanin – "A COMPARISON OF SOAP AND REST MPLEMENTATIONS OF A SERVICE BASED INTERACTION INDEPENDENCE MIDDLEWARE FRAMEWORK"