CHAPTER 3 RELATED WORKS 3.1 Service Orientated Architecture and EA

The researcher reviewed the SOA and its role in EA as the framework developed is also embodied in the principles of SOA and EA, in order to understand the strengths offered by such systems were studies along with the weaknesses, and tried to minimize the weakness in the framework developed, accordingly the following papers were chosen.

E-gov is visualized an public enterprise, and Service orientation is an essential prerequisite in achieving interoperability, protecting the legacy system investments and integration of old with new, and entering technologies and for transforming the electronic services to mobile, As the complexity of government
increases, the pain of development and maintenance of IS also grows. Governance, design patterns, and enterprise architecture are a part and parcel of any EA applications and hence this paper. “A Lifecycle approach to SOA Governance,” by Schepers, Lacob, & Van Eck, 2008 talks about the importance for applying governance to SOA. The authors propose a governance lifecycle with phases that cover strategy, organizational alignment, service portfolio, and policies. When we start to develop complex leaving aside simple WS based systems one must evolve standardised governance framework, to enable this this design patterns helps in providing solutions for recurring software design challenges. In the article

```
Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects (Schmidt, Douglas et al, 2013) examines SOA from the architectural patterns perspective. SOA based complex architected systems needs many artefacts to be created and maintained, as the IS grows and the complexity for maintaining artefacts also grows in geometric proportion in order to simplify the process complexities better frameworks are needed to manage the information, architectures, and the services that make up the system. A complex system like e-gov with varied artifacts, multitude of services demands a governance process to be in place. Enterprise Architecture (EA) is designed to address this problem of complexity. In 1987, the IBM systems journal carried an article written by Zachman named “A Framework for Information Systems Architecture.” Is considered as the bible of enterprise architects and it is Zachman who proposed the necessity of a framework “With increasing size and complexity of the implementation of information systems, it is necessary to use some logical construct or architecture for defining and controlling the interfaces and the integration of all of the components of the system” (p. 23). This has resulted in a number of frameworks or architectures that are in use today, viz

```

```

```

3.2 REST

```
Representational State Transfer REST is a widely accepted style of architecture invented and implemented by Roy T. Fielding of University of California, Irvine introduced in 1994 and published his Dissertation titled, “Architectural Styles and the Design of Network-based Software Architectures.” in 2000. Richard Taylor and Roy Fielding brought out an article during 2002 titled, “Principled Design of the Modern Web Architecture.” The article explained how the HTTP protocol is related to the REST WS, the conclusion of the authors in the article is that, “REST has served as both a model for design guidance and as an acid test for architectural extensions to the Web protocols”. In recent years, REST has been linked to REST services or RESTful services. REST is gaining in popularity, and it is considered a way to build simpler services, which are light weight and suited for resource crunched mobile environments. The main disadvantage of REST is its limitations in supporting frameworks that inflict its style and basic design protocols (Erenkrantz, Gorlick, Suryanarayana, & Taylor, 2007, p. 262). REST is not a technology, protocol, or a specification, and there is not an exact set of accepted compliance tests to determine if a system meets “RESTful” compliance. Though REST is not a technology, once should
understand that the services use HTTP not just as a transport protocol, but also as an application protocol. So it is through the understanding of the REST concepts, advantages and disadvantages, the choice of style of architecture of the APIs to be parked in the framework developed by the researcher was arrived. 3.3 Application Programming Interface (API) 3.3.1 What is an Application Programming Interface [API]? An API is a general term for inter application communications in a distributed environment, any kind of interface that enables one application to consume the functionality of another application. For the study purpose the term API may be defined as the public interface of a WS. A WS is just one of the forms of API: it accesses an API through a Web protocol (HTTP). There has been a recent rise in the number of public APIs being used today. Statistics from programablweb.com shows that there exists more than 10000 published API available for public consumption on various areas. Study by (DuVander, 2012) brings to light that the usage of APIs has doubled in 2011-12 alone. Another study by Vinoski and Pautasso in 2012 brought out a study report regarding the programmatic Web interfaces. It reveals the popularity of RESTful Web Services [Vitmar etal 2012]. Based on the statistics the researcher has acquired from programmableweb.com about 75% of the APIs were REST based and less than 25% SOAP based. While there may be many reasons why companies are choosing to implement one of the main reasons for massive REST APIs adoption by companies/applications is easy enablement of data collaboration (Jacobson, 2012). An API provides a medium for data collaboration; it provides standardised interface for governments to publish data as part of their open data initiatives and from the client perspective to obtain data through WS calls. 3.3.2 Public vs. Private APIs The numbers and trends in “public” APIs must not be confused with the vastly larger number of private APIs that are in existence (Jacobson, 2012). Jacobson points out that Many companies use APIs internally to allow their applications to reuse the same data and functionality and exposes such services and data to only partner companies, and the published API is just a tip of the iceberg. Organisations/governments expose the functionality of the services offered to the public by publishing their functionality through world via the internet. Documentation accompanying public API will define its terms usage rules, limitations etc, and provides developer with the parameters and options for consuming the API. Multiple revenue models is in use for API usage like pay per use, unit-based, and fermium (Jacobson, 2012). In the tiered model, there are different levels of usage. For instance a lower tier may cost less than a higher tier, but the lower tier will allow a much smaller number of API requests in an allotted timeframe. In the Pay as You Go model the API provider charges the client application based on the number of requests made to the API. In the Unit-based model, the client pays according to specific units of computing or service. And in the freemium model the client may use the API for free, but must pay for various types of additional services. For example, Google Maps uses a freemium model, in which they allow up to 10 thousand calls per day for free, and then charge for exceeding the terms of the plan (Jacobson, 2012). 3.3.3 RESTful APIs In RESTful Web services, the client makes an HTTP request to a specific URL, along with certain parameters set in either the URL, or in the request header. These requests are typically limited to four operations: GET, POST, PUT, and DELETE, as opposed to SOAP, which allows the developer to define any number of methods (Vaswani, 2010). Responses to the client take the form of a standardized HTTP response, returning a response code, as well as the response data to the client either as XML of JSON. As Vinoski points out as the REST APIs make use of HTTP an universal interface for all web based services (Vinoski, 2007). Client applications developers need to understand how to comply with the SP interface for invoking its resources using HTTP. The ability of REST WS to handle large amounts of concurrent requests when compared with SOAP WS makes REST a better choice for m-gov or e-gov applications which demands concurrent user access (Meng, 2009). 3.3.4 What is HATEOAS (Hypermedia as the Engine of Application State) There are
various interpretations and forms of REST, which was originally developed as part of a PhD dissertation of Roy Fielding (Jacobson, 2012). Fielding proposed HTTP protocol be used to allow computers to communicate by dividing URI namespaces into a set of resources (Fielding, 2000). There are many people in the industry who advocate a certain style and interpretation of Fielding’s definition of REST other group advocates that REST should embody the concept of HATEOAS. The principle dictates that instead of the SP defining a list of resources and actions the client may use, the functionality provided by the API in the form of additional URIs should be made available through an initial request to the root URI of the API. By enforcing HATEOAS principle or a clients requesting stale resources can be stalled. 3.4 Publish/Subscribe Model A request from consumer for up to date information from a conventional server will not be able to provide information because servers face tremendous overhead and there cannot be an appropriate method for effective and wide information diffusion amid working along with a big disseminated network. Pub/sub paradigm (Liu et al 2010) is a communication model that provides data and information propagation in a bigger mobile network. When it comes to the pub/sub infrastructure, the information dissemination is carried out as events and the users have to sign up their accounts as subscribers for their own choices of events. Just as the middle man in the event management system an event broker performing as a middleware dispatch events to the respective subscribers as per requests (Hung mole, 2001). Asynchronous and transparent nature of communication in pub/sub helps both service supplier and consumer work asynchronously through a sender and propose alterations to every subscriber through s single operator. Pub/Sub system in its basic model proffers a connection for service providers and consumers right through a set of groups or channels. It is through this channel the interested person get information and the publishers will transport the event to the respective subscribers. 3.4 .1 Subscription Schemes. Every consumer is different so is their interests, so the consumers may not react to all the announcements of the SP. Many subscription models are presently seen in pub/sub environments. Topic based and content based are considered as the most popular and widely used schemes. 3.4.2 Topic Based Topic based is a prominent scheme in the first generation schemes. The interest of a specific group or logical channels (baobe, 2005) related to a topic or the subscribers will announce subject of interest. Users can avail all events of a particular channel by subscribing it. The Topic based model has been proposed as a remedy in several pub/sub environments. CORBA notification service is one of the most mentioned systems (object mgmt. 2002). Baldoni and
Virgillito, 2005 has listed some systems which implement topic based scheme, they include Bayeux systems, SCRIBE and TIV/RV. Limited expressiveness is one of the major drawbacks of this scheme even if the SC

registers for a subsidiary set of events of a topic, the

SC would receive the entire event that published according to that topic. Organizing the events hierarchically is one of the solutions proposed by Eugster et al 2003].

3.4.3 Content Based

The Content-based scheme is a versatile paradigm when compared to

topic based scheme. In this scheme,

the subscriber has more command on

subscribing an event based on the original content of the event.

The scheme permits the SC to induce a

set of curtails in the form of condition in developing a query.

This enables SC to approach the events subscription in a complex fashion. 3.4.4 Messaging System

In a messaging system messages are managed

in a way a persistent database is managed by a database system. Messages are coordinated and integrated among the software components because software applications change from time to time. Transference of messages from one machine to another is done over the unreliable wireless networks. The Inherent limitation of wireless network makes the messaging system suitable to operate as it repeatedly tries to transmit messages until it has been sent.

Create, send, deliver, receive and process are the five steps in sending messages (hohpe 2004]. The
basic concepts in a messaging technology revolve around the key terms of message channel and routing messages. 3.5 Security and Privacy

Patel, Mohandes et al. has brought out several vulnerabilities of PKI based signature and encryption mechanisms, which was thought to be very secure, in their paper “Attacks on Web Services and mitigation schemes” [Patel & Mohandes, 2010]. Multiple critical security vulnerabilities of Open SSL the most popular Open Source SSL libraries in use was revealed by (Common Vulnerabilities and Exposures, 2014) even though OpenSSL's source code was open for peer reviews for several years. As the threat landscape is changing rapidly one should device resilient plants to thwart major security breaches to protect the data and ICT infrastructure. Securing RESTful Web services involves securing the data, and the messages transacted. Each of us is obliged to protect the privacy and integrity of data or information. A data under transmission should undergo filtering to avoid spiteful payload. The information exchange or communication must back the verification and access control to assure that the confidentiality of the communicating data is not compromised.

RESTful services depend on different add-ons that work on the platform HTTP, HTTPS when compared to WS- Security framework [40]. [E. Rescorla, 2000] is commonly used for confidentiality; however, it can only offer hop-by-hop security. Developers should make use of message level mechanisms for security. There is no prewritten standards to follow like in WS-*, but certain reference structures are followed by its worshipers. e.g. Amazon S3 service [3], which implements time stamps to avoid replaying the request. Different client side and server side filters are using to authenticate the content. HTTP succour basic digest oriented verification mechanisms [Franks 1999], in fact both of them have their own shortcomings [4]. Now a days, the current services delegate authentication mechanism and identity management.
to a third party. These services depend on a claim based verification practices. In recent times, the digital world has witnessed the emergence of technologies for supporting the verification for HTTP oriented services, e.g. OpenId [15] for federated identity, and OAuth 1.0 [23] for authentication and data sharing. These practices have lead to new forays of research. For instance, OAuth has decided to conduct a revision in October 2010, where the protocol writers are noticing the coding operations and depending on SSL to shield plain text exchange of verification tokens. They are selling off the security mechanisms to nullify the errors that in turn ease the programming process; but should be verified and validated in accordance with the research findings. A much touted protocol emerging recently is XAuth [29]. XAuth is an open platform for disseminating authenticated user services across the length and breadth of the web world, which always has multiple security problems.