Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

By

Alemtsehay Kebede

An Improved Technique for Enterprise Service Bus Data Transformation: The Case of ethio telecom

A Thesis Submitted as a Partial Fulfillment of the Requirements for the Degree of Master of Science in Telecommunication Engineering

Advisor: Mesfin Kifle (PhD)

Date: October 19, 2018

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

An Improved Technique for Enterprise Service Bus Data Transformation: The Case of ethio telecom

By

Alemtsehay Kebede

Approval by Board of Examiners

Chairman, School Graduate Committee	Signature
_____	_____
Advisor	Signature
Dr. Mesfin Kifle	_____
Examiner	Signature
Dr. Eng. YihenewWondie	_____
Examiner	Signature
Dr. Surafel Lemma	_____

Abstract

Service Oriented Architecture (SOA) is an integration architecture approach that is based on the concept of a service and addresses, the requirements of loosely coupled, standards-based, and protocol independent distributed computing. SOA used Enterprise Service Bus (ESB) to realize its principle. ESB is architecture to overcome the limitation of traditional architectures. One of its main functions is Data Transformation, which is the conversion process of data structure among heterogeneous systems during integration. Data transformation consists of four data transformation functions which are name and value transformation, attribute aggregation and splitting. Its performance depends on the data exchange format and transformation functions complexity. Most ESB platforms adopt XML-based format as their common data model, which is quite time consuming on data transformation. The performance of XML based format is affecting the real time communication performance, mainly as complexity increases.

The objective of this research is to analyze the performance of ESB data transformation functions and propose an improved technique for ESB data transformation data exchange formats. We used various research methods including literature review, informal interview and focus group discussion for gathering and analyzing relevant data.

An improved technique for ESB data exchange formats is proposed that uses both XML and JSON formats based on complexity matrix in Complexity Analyses Algorithm (CAA). CAA uses JSON format for high complexity and XML for low complexity level.

From the experiment result we have observed that JSON data format requires less time than XML data format while data complexities increases. So by implemented this improved technique we can enhance the performance of ESB data transformation.

Keywords--*Enterprise service bus; Data transformation performance; Data exchange format; complexity matrix, Complexity Analyses Algorithm*.

Acknowledgement

First of all, I would like to express my sincere and special thanks of gratitude to my advisor Dr. Mesfin Kifle, who has given me great help, excellent advice and has treated me with infinite patience throughout this thesis work. I also want to thank my husband for encouraged me along the way and advice. Beside him, I want to thank my Kids for their patience during the thesis work. My gratitude also goes to my family and friends for their help and best support particularly during this thesis work.

Finally, I want to give my gratitude to ethio telecom and Addis Ababa University for giving me the chance to study in this program.

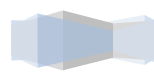
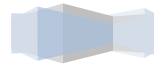


Table of Contents

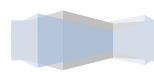
Acronyms

AJAX	Asynchronous JavaScript and XML
AMF	Action Message Format
API	Application Program Interface
B2B	Business-to-Business
BSS	Business Support System
CAA	Complexity Analyses Algorithm
CBE	Commercial Bank of Ethiopia
CBS	Convergent Billing System
CIM	Common Information Model
CIS	Component Interface Specification
CRM	Customer Resource Management
DIM	Data Model for Integration
DOM	Document Object Model
EAI	Enterprise Application Integration
EDI	Electronic Data Interchange
EIP	Enterprise Integration Patterns
EIS	Enterprise Information Systems
ERP	Enterprise resource planning
ES	Enterprise Systems
ESB	Enterprise Service Bus
ET	ethio telecom
HRMS	Human Resource Management System
HTTP	Hyper Text Transfer Protocol
HTTP	Hyper Text Transfer Protocol
IBM	International Business Machines
IEEE	Institute of Electrical and Electronics Engineers
IPCC	Internet Protocol Call Center
IPsec	Internet Protocol Security
IT	Information Technology

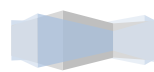
J2EE	Java Platform, Version 2 Enterprise Edition
JMS	Java Message Service
JS	JavaScript
JSON	JavaScript Object Notation
MOM	Message-oriented Middleware
P2P	point to Ponte
PHP	Hypertext Preprocessor
PHP	Hypertext Preprocessor
RAM	Random-Access Memory
RPC	Remote Procedure Calls
SAX	Simple API for XML
SAX	Simple API for XML
SCM	Supply-Chain Management
SDP	Session Description Protocol
SGML	Standard Generalized Markup Language
SIM	Subscriber Identity Module
SMSC	Short Message Services Center
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SRR	Service Registry and Repository
TEP	Telecom Expansion Project
UDDI	Universal Description, Discovery, and Integration
USSD	Unstructured Supplementary Service Data
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition
YAML	Ain’t Markup Language



List of Figures



List of tables



1. Chapter One: Introduction

1.1 Background

Rapid advances in industrial information integration methods have prompted tremendous growth in the use of enterprise systems (ES). Consequently, a variety of techniques have been used for searching ES. These techniques include business process management, workflow management, Enterprise Application Integration (EAI), Service-Oriented Architecture (SOA), grid computing, and others [1]. Today the organizations heavily rely on the technologies which are of different nature and the applications running in different business units. So the integration of these applications into a unified set of business process has emerged as a priority [2]. Due to this reason many organizations use ES to manage their business activity. In the past decade, ES has emerged as a promising tool used for integrating and extending business processes across the boundaries of business functions at both intra-organizational and inter-organizational levels [1].

Enterprise Application Integration (EAI) is aimed to make applications be able to communicate with each other. It is used to unrestricted sharing of data and business processes among any connected ESs and data sources in the enterprise [3, 2]. The paradigms for EAI have evolved over time, from point to point (P2P) solution, to message brokers (hub-and-spoke), and to enterprise services bus (ESB)[4].Integration is necessary to takes into account the business strategy as defined from the enterprise vision, the business process definition, and operation of interoperable enterprise systems are supported by relevant and efficient IT infrastructure. Service Oriented Architecture (SOA) plays a crucial role in the architectures of most commonly used integration platforms of today [5].

Service Oriented Architecture (SOA): is a technology that helps integrate heterogeneous systems, as it provides a data bridge between incompatible technologies [6]. It is a component model, and it can relate different functional units called service of the application through these well-defined interfaces and contracts between services. The interfaces are defined in a neutral manner, which should be independent of the hardware platforms, operating systems and programming languages. This makes different services that are in various systems interact in a uniform and common way. ESB is specific mechanism to achieve SOA [7].

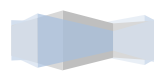
Enterprise Service Bus (ESB): is an improved architecture to overcome the drawback over P2P and Hub-Spoke architectures and plays a critical role in connecting heterogeneous applications and services in a SOA [8].Those P2P and Hub-Spoke are traditional architectures which used for integrated different systems. These architectures have scalability issues and introduce a single point of failure in the

network. But ESB is used to integrate many services from many departments by using their owned platform and technology of information system. It can be the solution of n-to- n integration problem and it strongly supports the implementation of SOA [6]. It is a product combined with traditional middleware technology, Extensible Markup Language (XML), Web services and so on[9, 10]. Currently there are different types of ESB solution some of them are: Microsoft BizTalk, IBM Web Sphere, SAP Net Weaver and Oracle Fusion Middle Ware integration [5].

In ethio telecom IBM Web Sphere ESB was implemented as middleware technology to realize SOA principle. Web Sphere ESB delivers its infrastructure to connect applications that have standard-based interfaces. It can be obtained as a stand-alone product and included in web sphere process Server [8]. ESB provides different functions such as Routing, Data Transformation, Protocol Transformation and others; this research is focus on the issue that related with data transformation function on real time communication.

Data transformation is the conversion of data format and type among heterogynous transformation is the conversion of data format and type among heterogynous systems during systems integration [9]. The data transformation has four main transformation functions they are called [10]: Name transformation, Value transformation, Attribute aggregation and splitting. The occurrence of those functions with in single message is determining the complexity level of data transformation [10]. For example massage "A" required converting two different functions which are converting Full-Name= Alem Mamo to Name = Alem Mamo and Birth date = 9/12/2003to Age = 15. For the second example in addition to the previous example add attribute aggregation and splitting function which means transformation doing from First-Name = Alem, Second-Name = Mamo to Full-Name= Alem Mamo and vice versa for splitting. According to [10] the second example is more complex than the first one and the performance of data transformation is dependent on the data exchange form at related with function complexity level. This complexity level is determined by the appearance of data transformation function within single message.

There are different data transformation technologies, the widely used are, Extensible Markup Language (XML) and JavaScript Object Notation (JSON). XML is a subset of the Standard Generalized Markup Language (SGML) and evolved as a result of the complexity of SGML. It is considered the 'holy grail' of computing due to its universal data representation format. The intent of an XML document is self-evident and embedded in its structure [11]. JSON is designed to be a data exchange language which is human readable and easy for computers to parse and use [11]. However, most ESB platforms adopt



XML-based model as their common data model, which is quite time consuming on ESB data transformation [10].

1.2 Statement of the Problem

Ethio telecom uses IBM Web spheres ESB as a middleware to integrate heterogeneous systems. This middleware is used to integrate systems within business support systems (BSS) and BSS with third party systems at real time communication. ESB is provides different functions for integrate heterogeneous systems; one of them is data transformation. The performance of ESB data transformation is affect by the technology that using for data exchange from one system to the other and data transformation functions complexity level. Those factors are directly related with the performance of real time communication.

The company business activities mainly depend on real time communication and some of the processes are: *Subscriber Identity Module (SIM) card sales, SIM replacement, SIM transform, offers change, Billing* and others. The completeness of those processes is affecting the company with two perspectives, from the company point of view it assures telecom services available and from the customers point of view also it is the key indicator to provide the service.

IBM Web spheres ESB uses XML base format for exchange data among heterogeneous systems, this type of data model is time consuming than the other data model. So the impact of this data model is affecting the real time performance requirement, which leads to customer dissatisfactions and revenue loss.
For example: during SIM card sale, first insert necessary information in Customer Resource Management (CRM) system and then message request is sent to Convergent Billing System (CBS), those systems are doing their own tasks after the processes are completed they exchange notification to assure process completion and the services availability. But one of the factors that affect the efficiency of data transformation task is the performance issues that appear due to XML based format.

- Even if XML Web Service have some advantage, it is not enough to enable SOA adopters to use it effectively in real-time business systems, as it still suffers from number of problems such as low performance, bad utilization for hardware resources, and high network latency [12]. So this research is focus on the performance of ESB data transform function which appears due to data exchange format on real time communication.

1.3 Objective

1.3.1 General Objective

The general objective of this research is to analyses the performance of ESB data transformation functions in real time communication and optimized data transformation process by using an improved technique for ESB functions complexity based data exchange format by applying complexity matrixes on Complexity Analyses Algorithm in the case of ethio telecom.

1.3.2 Specific Objectives

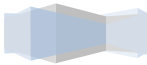
The specific objectives of this research are:

- To assess and analyze the performance of ESB data transformation function by focusing transformation functions and data exchange format.
- To compare and analyze widely used data exchange format which are XML and JSON formats related with required time for parsing.
- To improve the technique for ESB data transformation by using complexity matrix and complexity analyses algorithm that use XML and JSON formats at different complexity level.
- To evaluate the proposed technique.

1.4 Method

This research uses different method to gather and analyze the data and finally develop an improved ESB data exchange format for data transformation function to improve the performance of real time communication. The methods are:

- Literature review: different published papers, white papers, books and company design document such as low level design, high level design will be used.
- Interview and focus group discussion (informal): to gather information about the performance of implemented ESB data transformation functionality related with data exchange format. And identify the issue that relate with real time communication.
- Data analyses: By using and comparing different company documents and the- state-of-the-art identify the gap and proposed the solution.



- Evaluation: finally perform evaluation by using java programing to show the objective of proposed solution is achieve or not.

1.5 Scope and Limitation

There are different factors that affect the performance of real time communication some of them are: Bandwidth, Network, and Data model type. But this research is focused only on execution time related with the data exchange format in ESB data transformation functionality.

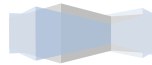
1.6 Significance of the Study

In the company many time sensitive business transactions are processed every day. Those transactions are the key factor for the company core activity which is telecom service availability, maximize the company revenues and manage the customer's requests. So using XML based data exchange format in ESB data transformation has an impact on performance of real time communication the impact of this pass to company business activity.

Therefor the significance of this research is by using time sensitive data exchange format optimized the performance of ESB data transformation function, minimized the issues which create due to the performance of data transformation which are related with data exchange format. The result of optimization is bringing an improvement on the company business activity; service availability and maximized company's revenue.

1.7 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 – Literature Review: describes the core concepts of the research. Chapter 3 – Related work: In this part will discuss the core ideas of the research by using the related scientific works which are relevant with those concepts. Chapter 4 The Proposed Technique: In this chapter will discuss ESB implemented in ethio telecom and proposed solution that use to improve data transformation functionality performance. Chapter 5 – Evaluation: the proposed solution will be testing and discuss the result based on the objective of the research. Chapter 6 – This chapter is all about Conclusion, Recommendations and Future Work, Reference and Appendix.



2. Chapter Two: Literature Review

2.1 Introduction

This chapter is focused on the basic concepts of the research by using different scientific research papers, articles and books. It consists of seven sections each of them are focuses on certain area: Section 2.2 Enterprise Application Integration Section 2.3 Enterprise Service Bus 2.4 ESB Data Transformation Functionality and Logic. Section 2.5 Service Oriented Architecture. Section 2.6 Web Service. Section 2.7 Data exchange format

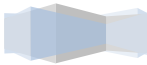
2.2 Enterprise Application Integration

Due to the rapid growth of business world and organizations imposed to depend on technology and need of the integration of the disparate applications is in high demand. With the development of information technology, over the past several years, there have been numerous distributed computing models in the EAI domain, including Message-oriented Middleware (MOM), EAI, Web services, SOA and ESB. Before the appearance of ESB, many companies have adopted tradition architecture. ESB is a middle ware technology use to overcome the limitation that appears in traditional architecture [14].

2.2.1 EAI Approaches and Techniques

EAI is the combination of processes, software, standards, and hardware resulting in the integration of two or more enterprise systems allowing operation as seamless manner [2]. As an enterprise consists of many heterogeneous systems, organizations had started integrating applications in the simplest form, connecting each application with every other application. But, in this situation the numbers of applications and complexity have direct relation with each other. While the numbers of applications increase, the complexity of integrated also increases [2]. The following section shows the evolution of EAI from the most traditional P2P integration to the most popular bus architecture of integration which is ESB [2]. EAI can also be categorized according to its design structure. The following lists are the main EAI architectures [2]:

- Point-to-Point topology
- Hub-Spoke topology
- Bus Topology



2.2.1.1 Point-to-Point

In P2P integration model, a unique connector component is implemented for each pair of applications or systems that must communicate. This connector handles data transformation, integration, and any other messaging related services that must take place between only a specific pair of components. When used with small infrastructures, where only two or three systems must be integrated, this model can work quite well, providing a lightweight integration solution tailor-made to meet the needs of the infrastructure. However, as additional components are added to an infrastructure, the number of P2P connections required to create comprehensive integration architecture begins to increase exponentially [13]. The figure 2.1 illustrate that the interconnection among the systems by using P2P architecture.

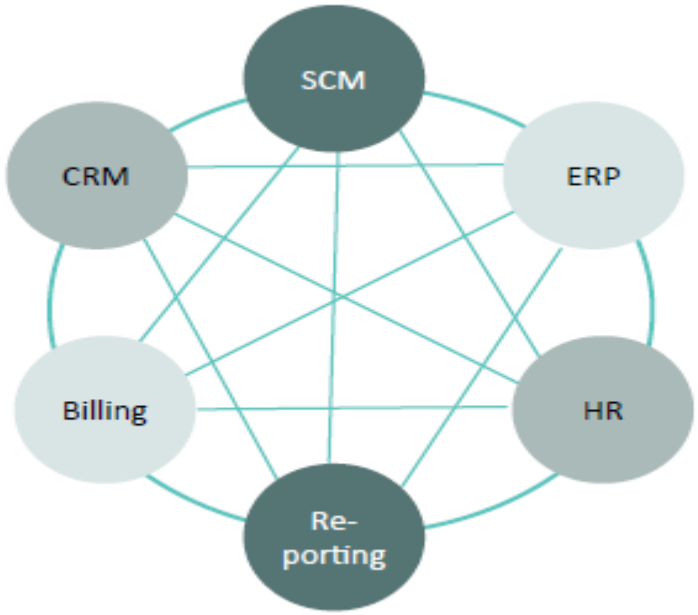


Fig 2:1 Fully Meshed Point-to-Point Connections N (N-1)/2 [13]

The main advantages of this connection are [2]:

- Its ease and speed of implementation for few applications
- Deployed relatively cheaply and rapidly.

The disadvantages of this connection are [2]:

- Tightly coupled
- Causes of spaghetti
- Difficulty to manage

- Become costly while the connection in large number of applications.

Scalability issue, it need n*(n-1)/2 number of connection for fully mesh connection. N represent is number of systems.

To avoid the complexity and unreliability of integrating complex infrastructures using the P2P approach, EAI solutions use various models of middleware to centralize and standardize integration practices across the entire infrastructure. Rather than each application requiring a separate connector to connect to every other connector, components in an EAI-based infrastructure use standardized methods to connect to a common system that is responsible for providing integration, message brokering, and reliability functionalities to the entire network [13].

2.2.1.2 Hub-Spoke

This approach involves a HUB - a central integration engine that resides in the middle of the network, and facilitates message transformation, routing, and any other inter-applications functionality. All the communication between applications must flow through the hub, allowing it to maintain data concurrency for the entire network. The figure 2.2 illustrate that the connection between systems and hub.

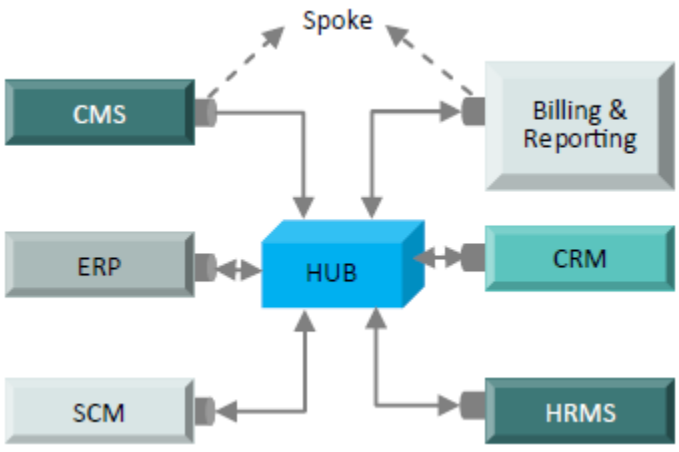


Fig 2:2 Hub and Spoke Connection [13]

The main advantages of this connection are [13]:

- Loose coupling between applications, which means that applications are able to communicate asynchronously.

- Less repetitive configuration i.e. all integration configuration to be accomplished within a central repository.

The disadvantages of these connections are [13]:

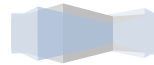
- HUB becomes a single point of failure for the network.
- Under heavy loads, the broker can become a bottleneck for messages.
- Broker models are often heavy weight, proprietary products, aimed at supporting a specific vendor.

2.3 Enterprise Service Bus

An ESB is a key enabler for a SOA as it provides the capability to route and transport service requests from the service consumer to the correct service provider [15]. It is a highly distributed, event-driven, enterprise SOA that is geared toward integration. And it is standards based integration platform that combines messaging, Web services, data transformation, and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications across extended enterprises with transactional integrity. An extended enterprise represents an organization and its business partners, which are separated by both business boundaries and physical boundaries. In addition to that an ESB is suitable for any project, no matter how large or how small [14].

In a complex enterprise computing environment, there are several applications interact with each other if they using P2P model, the relation between applications become complex and will form a net. This will bring high maintenance costs and make the reuse of the applications difficult. To overcome the limitation of traditional integration architectures using ESB is more suitable for complex enterprise. ESB is emerging as a middleware infrastructure component that supports the implementation of SOA within an enterprise [15]. It is the specific mechanism to achieve SOA, and the agency to achieve intelligent integration and management among services [7]**.** The need for an ESB can be seen by considering how it supports the concepts of SOA implementation by [15]:

- Decoupling the consumer's view of a service from the actual implementation of the service
- Decoupling technical aspects of service interactions
- Integrating and managing services in the enterprise



It has a flexible connectivity infrastructure for integrating applications as service. The interface of Web Services makes ESB easier to connect with heterogeneous applications. The Figure 2.3 illustrates the connection between service consumer and provider through ESB [7].

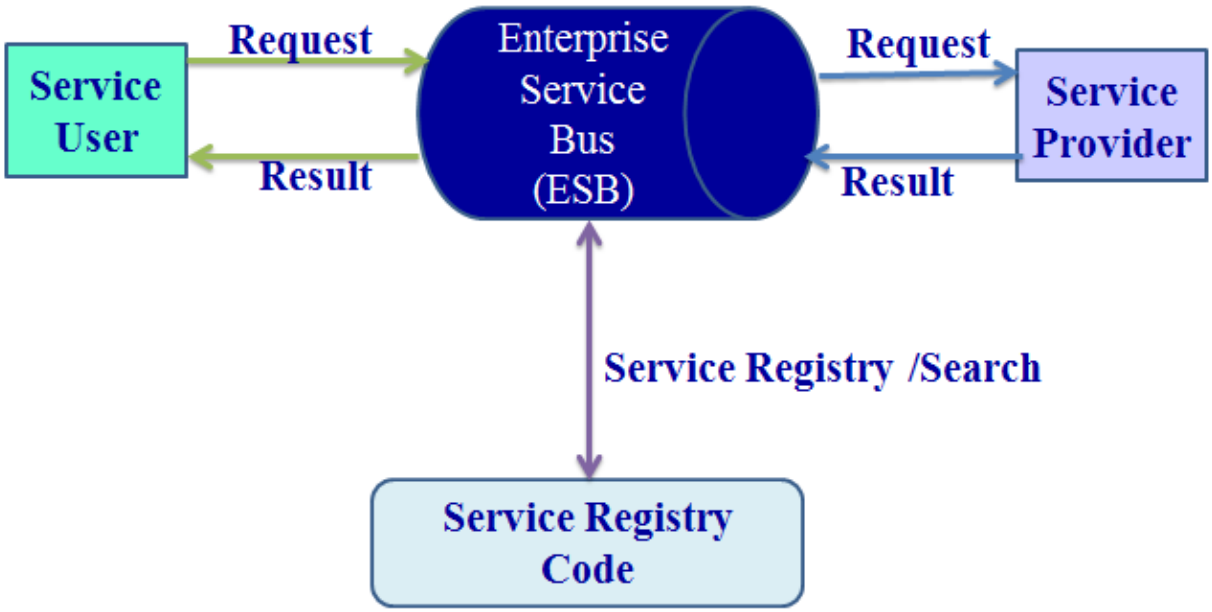


Fig 2:3 The Model of ESB Based on SOA [7]

In the figure service user is represent an application which sent the request and reserved the response message by using its data format and type. Service Provider is representing the applications which have received the request and provides the respond by using its own data format and type. One of ESB function is data transformation, while different applications exchange message. Service Registry and Repository (SRR) is a key enabler for SOA governance. SRR provides registry functions that support publication of metadata about the function. The requirements and semantics of services allow service consumers to find services or to analyze their relationships. In addition, it provides repository functions to store manage and assign a version number to service metadata. It also enables governance of service definitions by providing the following functions [16]**:**

- Control access to service metadata
- Model life cycle of service artifacts
- Manage promotion of services through phases of their life cycle in various deployment environments

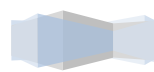
- Perform impact analysis and communicate changes to the governed service metadata

The ESB can allow users to achieve a low-cost, standards-based Web Services arrangement tools, and construct healthy SOA above this. It supports the services, the message in the heterogeneous environment and has an appropriate level of service and manages ability [9].

ESB simplifies the task of both service consumers and service providers by adding a layer of abstraction that shields the consumers and the providers from having to worry about the specifics of message format or message transport. It is provided different functions those used to create smooth integration among heterogeneous systems. Some of them are: Data Transformation, Content Based Routing and Virtualization [14].

- Virtualization: virtualization, or proxy, is one of the core capabilities of an ESB. In this scenario, the ESB received an incoming request and forwarded it the real Web Service. There was no other processing done by the ESB in this scenario. The ESB received the request from the client and forwarded it to the Web Service. The web service received the payload, appended a string to the payload, and sent it back to the ESB. The ESB, in turn, returned this payload to the client [14].
- Data Transformation: is the ability to convert the structure and the format of the incoming business service requests to the structure and format expected by the service provider. It is another core feature of an ESB. The ESB has the capability to take an incoming request and transform the message payload before sending it to the end Web Service. In this scenario, the ESB got a request from the client and transformed the message payload using XSLT. It then forwarded the message to the Web Service [14].

- Content Based Routing: The ESB has the capability to route the incoming requests on a single endpoint to the appropriate service. The ESB can look at a wide array of things like the message content or the message header to determine where the request should be routed. In this scenario, the ESB received the request from the client and inspected the payload for a keyword to determine the Web Service to which the request should be sent. The Web Service received the payload, appended a string to the payload, and sent it back to the ESB. The ESB, in turn, returned the payload to the client [14].



2.4 ESB Data Transformation Functionality and Logic

Various applications may not agree on the format for the same conceptual data. The sender formats the message one way, yet the receiver expects it to be formatted another way. To reconcile this, the message must go through an intermediate that converts the message from one format to another [19]. Data transformation function is one of the critical parts to facilitate such kinds of thing in ESB. There are four main data transformation functions in ESB they are called [10]:

- Data object's name transformation
- Data object's value transformation
- Aggregation of two or more data objects
- Data object's splitting

Figure 2.4is illustrating data transformation task with in two different application A and application B.

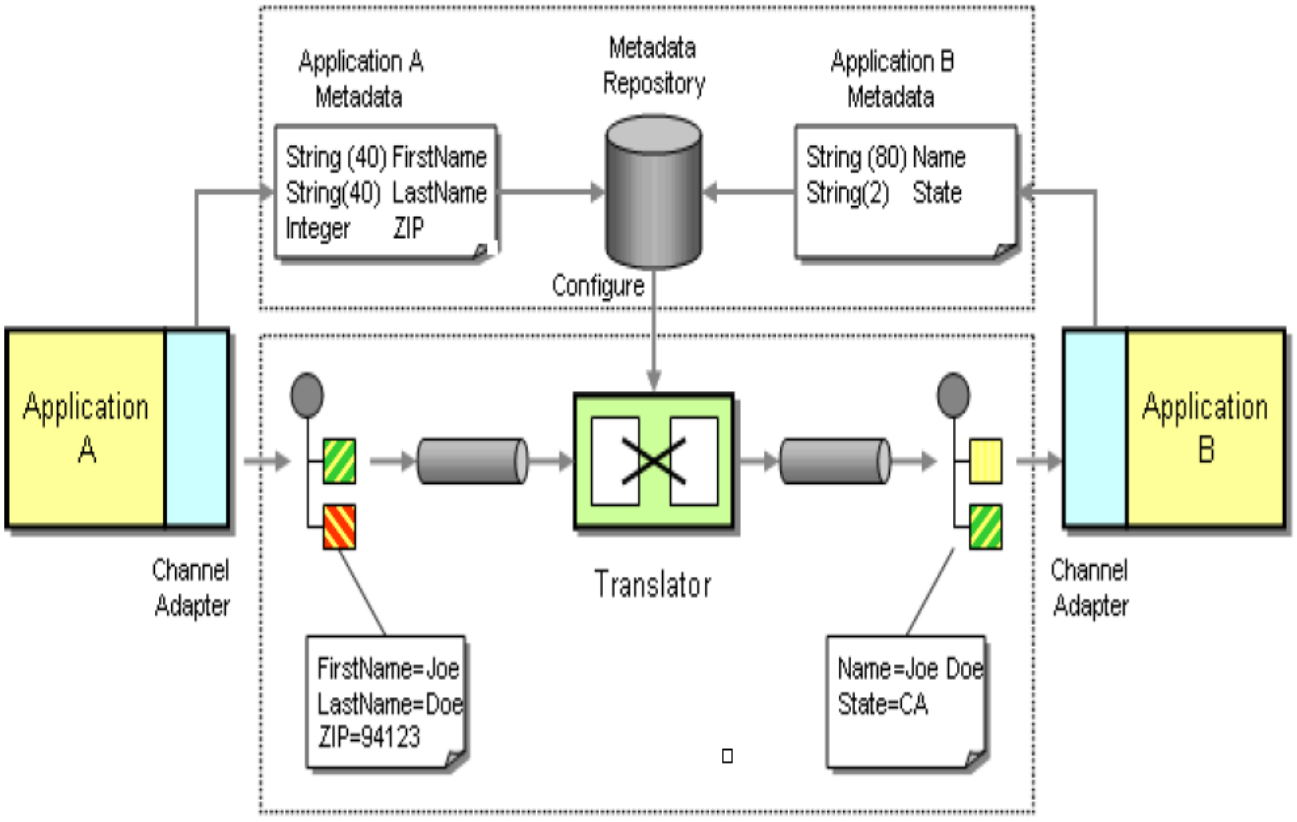


Fig 2:4 Data Flow Between Two Systems [28]

For example, Figure 2.4shows integration between two applications that need to exchange customer information. Each system has a slightly different definition of customer data. Application A stores first

and last name in two separate fields whereas Application B stores it in one field. Likewise, Application A stores the customer's ZIP code and not the state while Application B stores only the state code. Messages flowing from Application A to Application B have to undergo a transformation so that Application B can receive data in its format [28].

According to [10] the performance of data transformation is depending on data exchange format type and data transformation complexity of the data that transform from one system to the other through ESB. As we mention in Section 2.5 data complexity is determine by the appearance of functions in a single message which is the factor that affect the performance of data transformation. While the complexity of the data increase the required time also increase based on XML data exchange format.

2.5 Service Oriented Architecture

SOA is an integration architecture approach that is based on the concept of a service. The business and infrastructure functions that are required to build distributed systems are provided as services that collectively, or individually, deliver application functionality to either user applications or other services. A *service* is commonly defined as any function that carries out a business task that can be offered to an external consumer. The consumer invokes the service to use the business function that it provides. In addition to implementing a business function, a service in an SOA must also meet the following requirements [16]:

- Provide a function that is aligned to a business need
- Be defined by explicit, implementation-independent interface
- Be invoked through communication protocols that stress location transparency and interoperability

It provides an advantage over earlier approaches to integration because it focuses on business rather than technical requirements. In an SOA, services must offer functions that are aligned to business needs. Simply exposing IT functions by defining the functional interface using, for example, a web service does not make that function an SOA service if the function does not align to a business task. A company might have developed multiple web services; however, if they were developed ad hoc to meet point-to-point integration requirements, they probably do not offer a reusable business function and thus are not SOA services [16].

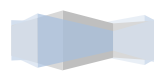
For example, a web service that is defined simply to solve a technical integration problem, such as to synchronize data between two systems can be valuable to meeting a specific integration requirement for the two systems, but it is not a good example of an SOA service. In this case, the service will be too tightly coupled to the specific requirements and details of the interaction between the two systems rather than offering a function that meets the business-level need and can be used and reused flexibly by multiple business processes rather than specific systems [16].

SOA specifies that within any given architecture, there should be a consistent mechanism by which services communicate. That mechanism should be loosely coupled and should support the use of explicit interfaces [17].

SOA is an emerging approach that addresses the requirements of loosely coupled, standards-based, and protocol independent distributed computing. Typically business operations running in an SOA comprise a number of invocations of these different components, often in an event-driven or asynchronous fashion that reflects the underlying business process needs. To build an SOA a highly distributable communications and integration backbone is required. This functionality is provided by the ESB that is an integration plat form that utilizes Web services standards to support a wide variety of communications patterns over multiple transport protocols and deliver value-added capabilities for SOA applications [25].

By adopting an SOA approach and implementing it using supporting technologies, we can build flexible systems that implement changing business processes quickly and make extensive use of reusable components [15]. Figure 2-3 illustrates a company that wants to implement a new business process to support customers who are placing orders from a Web site [16].

The basic requirement of a registry and repository is to provide a simple storage location for service specifications. However, the technology used to implement them, such as the registry and repository component in Web Sphere ESB Registry Edition, is often more sophisticated than just containing metadata about services. It provides features to help analyze relationships between consumers and providers and provides support for governance around service adoption and versioning and impact analysis [16].



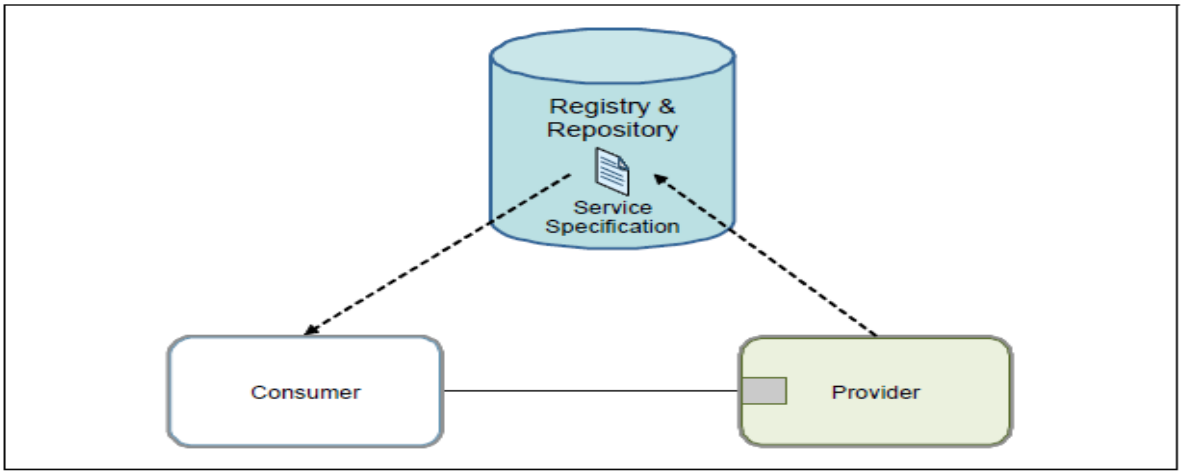


Fig 2:5 Service Registries [16]

Function of both a registry and a repository are explained as follow. The repository allows users to store, manage, and query content of documents that hold service metadata descriptions (WSDL, XSD, WS-Policy, SCDL, or XML documents). The repositories stores the documents that contain service metadata and also provide a fine-grained representation of the content of those documents (for example, ports, or port Types in WSDL documents) [16].

The repository also provides registry functions for populating registered service declarations and elements of the derived content models with user-defined properties, relationships, and classifiers. A rich query interface uses these user-defined attributed when users want to find a service endpoint, interface description, or other metadata about a service [16]. Organizations spend a considerable amount of time and money trying to achieve rapid and flexible integration of IT systems. This activity usually includes the following objectives [16].

- Increase the speed at which businesses can implement new products and processes, change existing ones, or recombine them to deliver new value.
- Reduce implementation and ownership costs of IT systems and the integration between them
- Simplify the integration work that is required by mergers and acquisitions
- Achieve better IT use and return on investment
- Implement business processes at a level that is independent from the applications and platforms that are used to support those processes.

SOA is an approach that helps to achieve this rapid flexible integration. It provides the following value [16]:

- Reduces IT constraints on the business to increase the company's responsiveness to change.
- Reduces IT costs by increasing reuse and removing redundancy and duplication in IT resources.

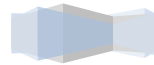
A service encapsulates a discrete business function and provides a clearly defined interface that makes it easily reusable. Applications are built by invoking services, and services can be chained together to form more complex functions. This flexibility allows companies to quickly implement business processes that meet changing business needs [16].

ESB is the specific mechanism to achieve SOA, and the agency to achieve intelligent integration and management among services. It develops from SOA, and combines a variety of technologies [7]. Web services are the preferred implementation technology for realizing the SOA promise of maximum service sharing, reuse, and interoperability [25].

2.6 Web Service

Web services are connect programs to each other across distant points on the global map, transporting large amounts of data more efficiently and cheaply [17]. It enables businesses to connect applications to other business applications, to deliver business functions to a broader set of customers and partners, to interact with marketplaces more efficiently, and to create new business models dynamically [15].

One of the main aims of Web services is to provide a loose coupling between service consumer and service providers. While this is limited to a certain extent by a requirement for the consumers and providers to agree on a WSDL interface definition, Web services have been created with significant flexibility with regard to the location of these Web services. Figure 2.6 shows how the Web services interaction model has been designed with this form of loose coupling [15].



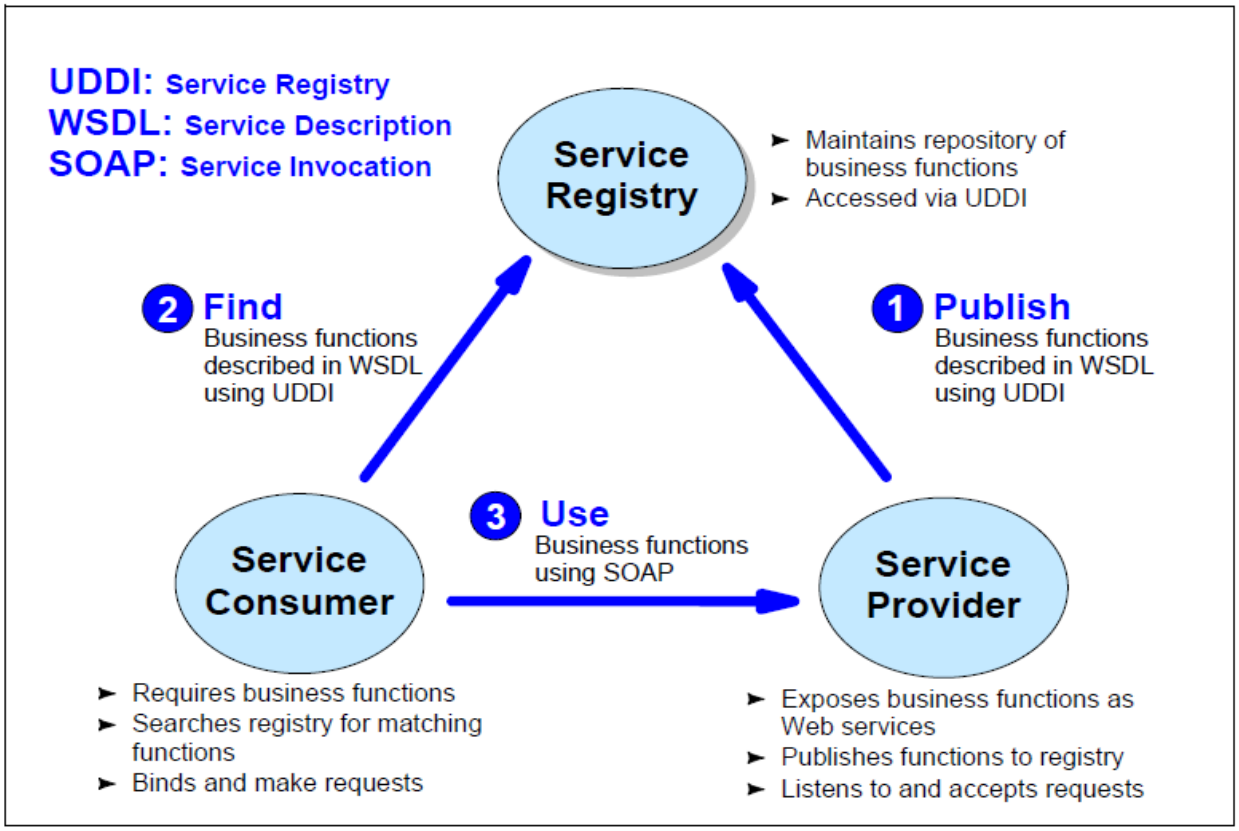


Fig 2:6 Basic Web Service Integration Model [15]

The interactions work as follows [15]

1. The service provider publishes some WSDL defining its interface and location to a service registry.
2. The service consumer contacts the service registry in order to obtain a reference to a service provider.
3. The service consumer, having obtained the location of the service provider, makes calls on the service provider.

As per [26] web services and SOA reduce complexity of enterprise application eco-systems by encapsulation and minimizing the requirements for shared understanding by defining service interfaces in an unambiguous and transparent manner. Also Web services enable just in time integration and interoperability of legacy applications.

Web services are using an XML document created in the form of a message, a program sends a request to a Web service across the network and options all, receives a reply, also in the form of an XML document. Web services standards define the format of the message, specify the interface to which a message is sent, describe conventions for mapping the contents of the message into and out of the programs implementing the service, and define mechanisms to publish and to discover Web services interfaces. Web services transform XML documents into and out of IT systems. Web services require the use of several related XML-based technologies they are called XML, WSDL, UDDI and SOAP [17].

XML: the basic foundation on which Web services are built provides a language for defining data and how to process it. XML represents a family of related specifications published and maintained by the World Wide Web Consortium (W3C) and others.

WSDL (Web Services Description Language): an XML-based technology, defines Web services interfaces, data and message types, interaction patterns, and protocol mappings.

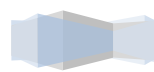
SOAP (Simple Object Access Protocol): a collection of XML-based technologies defines an envelope for Web services communication map able to HTTP and other transports and provides a serialization format for transmitting XML documents over a network and a convention for representing RPC interactions.

UDDI (Universal Description, Discovery, and Integration): a Web services registry and discovery mechanism, is used for storing and categorizing business information and for retrieving pointers to Web services interfaces.

These technologies can be used in many ways. Web services can also be used for business-to-business (B2B) integration, connecting applications run by various organizations. It can also solve the broader problem of enterprise application integration (EAI), connecting multiple applications from a single organization to multiple other applications both inside and outside the firewall. In all these cases, the technologies of Web services provide the standard glue connecting diverse pieces of software [17].

There are different parameters that play a major role in determining the memory and bandwidth performance of a SOA application: some of them are marshalling and unmarshalling [19]. Marshalling is a wider concept and a superordinate grouping word. Marshalling can be done by serializing and the reverse process is called unmarshalling [18].

The web services' performance is largely dependent on three related factors. One of the key factors has to do with the processing capability of the object marshalling and unmarshalling tasks in the server and those of the client end points. Another key factor is the latency of marshalled object transfer, which



depends on the verbosity and the resulting length of the used representation format. The last of the key factors is the performance of the underlying service or business logic, which is usually stabilized and solidified [18].

The effective marshalling throughput was calculated as a quotient of the complete effective data length in bytes and of the measured time spent in seconds on serializing it. The effective unmarshalling throughput *TP* is the effective length of the input serialization result in bytes divided by the measured time in seconds spent on unmarshalling it as in, where *L* is the effective length and *T* the measured time in seconds [18]. The equations of through put express in the following manner:

$$TP = L/T \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2, 1)$$

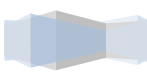
2.7 Data Exchange Format

Currently different type of technology is used to exchange message or to exchange request and respond data from consumer to provide, some of them are called: JavaScript Object Notation (JSON), XML, Action Message Format (AMF), and Ain't Markup Language (YAML) [20, 21]. YAML and JSON are two light-weight data interchange formats [20] and JSON and XML are two mostly used data interchange formats with unique purposes [11, 22]. In the next part we will discuss those four data exchange format.

YAML is emphasizing on its design as a data storage format. It is a light-weight human readable serializing language primarily designed to be easy to read and edit. By adding a simple typing system and aliasing mechanism upon the three most common data structures used when serializing (hashes, arrays and strings) it forms a language which is easy to use, while still including more complex features [20].

Action Message Format (AMF): is a binary format that is used for serialization of objects and sending messages between the client and the remote service. Action Script 3 languages has classes for encoding and decoding of the AMF format. So today there is AMF support for platforms written in Java, PHP, .NET and other languages [21].

XML is the most widely used data exchange format comparatively than the new JSON format by their fundamental characteristics. It is the most used technology for applying SOA because of easy to use and allows high interoperability. It is a language used for creating user-defined markups to documents and



encoding schemes. It does not have predefined tag sets; each valid tag is defined by either a user or through another automated scheme and user-defined hierarchical data format [22].

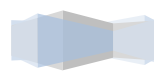
XML is the actual standard format for data exchanging in the context of web service applications [25].It is easy, flexible text based data exchange format obtained from Standard Generalized Markup Language (SGML). The primary uses for XML are Remote Procedure Calls (RPC) and object serialization for transfer of data between applications [21].

An XML document forms a branched structure that starts with the root element and must be "well-formed". The greatest strength of this model is the strict structure definition that is why it can be implemented for data validation[22].On the other hand, the XML document is, for many object-oriented languages, difficult to parse and convert into a structure or object with which they can work. There are many technologies for processing XML document such as: Document Object Model (DOM), Simple API for XML Simple API for XML (SAX), XPath and XSLT. The use of closing tags (<id> 15 </id> versus JSON's "id":15) make XML too comprehensive and partially redundant, which ultimately means a longer document reading time [22].

The XML document, because of the possibility of expanding XML attributes and CDATA sections, can store all possible data types. It includes the ability to transfer the structure and formatting of the document. This makes it much more flexible, but also harder to read. Flexibility is good for transferring documents, which can contain images, graphs, text, etc. But it is not suitable for simple data transmissions, because it has redundant and unnecessarily complex [22].

JavaScript Object Notation (JSON) is a key-value style lightweight data exchanging format. It is widely used format for interchanging data on the web after XML [23]. For its simplicity, JSON makes it easy for human to read and write and for computers to generate and parse [11, 23]. It is a native data form for JavaScript, which means no special APIs or jars are needed to process JSON data [23]. JSON also is language-independent, meaning that the specification is not tied to any specific programming language. The design incorporates data types common across most modern languages [20].

It is directly supported inside JavaScript and suited for JavaScript applications; thus providing significant performance gains over XML, which requires extra library to retrieve the data from DOM object. It is estimated to parse up to one hundred times faster than XML in modern browsers, but despite its claims of noteworthy performance, arguments against JSON include lack of namespace support, lack of input validation and extensibility drawbacks [11].



From those the above data exchange formats XML and JSON are mostly used format. XML and JSON as data storage and transmission formats have their strengths and drawbacks, which determine their usefulness for certain purposes. To transfer documents with a lot of different data types and elements, XML is the ideal choice. JSON is better suited for dynamic web applications and simple data transmissions. JSON performance speed is greater than XML's because of its simple structure and ease of access to data. JSON will not fully replace XML in the area of the Web. XML, because of its rich features, has its place in the transfer and validation of documents. JSON is better suited to data-interchange and should be used instead of XML in data transmissions between a server and web application [22]. JSON is the one of the tool for transferring simple data because data is stored in strings and records, while in XML data is stored in a treelike structure; it is not the most ideal structure for programming languages to understand [23]. The below section discuss about the main features of both XML and JSON data exchange formats and compare them vis-à-vis each other [19].

1. Industry Adoption
   - XML has been the industry standard for around a decade and has a lot of supporting frameworks and standards to govern the implementation.
   - JSON on the other hand is relatively new on the block and does not have standards like schematrons, XSDs, to govern its implementations.
2. Human Readability
   - XML documents are easy to read for humans.
   - JSON is highly cryptic due to the use of parenthesis delimiters.

   Metadata
   - XML has a big overhead in the form of tag metadata.
   - JSON has minimal metadata making it compact but a loosely defined format.
3. Supporting Frameworks
   - XML is supported by a majority of framework implementations (API) across the industry.
   - JSON lags behind but is catching up very quickly in terms of support adoption.
4. Extensibility
   - XML allows you to store any data type. Data element attributes allow additional flexibility.
   - JSON is limited to only storage of classical data like text and numbers.
5. Ease of Mapping

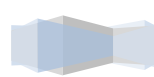
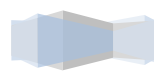


Table 1 XML Strengths over JSON [24]

JSON	XML
There is no grammar support and that's why it is difficult to communicate and enforce interface contracts	While XML have XML schema and Document Type Definition which can be used to define grammar rules
Extensibility is not good as namespaces are not supported	Very strong support for namespaces, schema have more extensibility options
Development tools support is very limited as it is newly introduced	As XML is in the market since long time there for is supported by most of the development tools
JSON is very narrow focused as it is used only for Remote Process Call(RPC), mainly with JavaScript Clint	XML is very broad focused, it can be used for Remote Process Call (RPC), Electronic Data Interchange (EDI), Metadata etc.
Very limited support for web services associated stuff (products).	huge hold of web services related products

Table 2 JSON Strengths over XML [24]

JSON	**XML**
Completely programmed technique for de-serializing and serializing JavaScript objects, with very little coding.	JavaScript code will be written by developer to serialize and de-serialize to and from XML
Most of the browsers have enough support of JSON.	All new browsers have built in XML parser but it could be a bit tricky when it comes to cross-browser XML parsing.
The format is very concise due to having name/value Pair-based approach.	Because of tags and namespaces the format is Very lengthy.
de-serialization is very speedy in JavaScript	De-serialization is slower in JavaScript
Most of JavaScript libraries and AJAX toolkits have good support of JSON	AJAX toolkits don't have strong support for it.
Having simple API for JS and more other languages	The APIs are very complicated



- XML is document oriented (Tree Structure) making it difficult to map it to objects in the OOP paradigm (Graph structure).
- JSON is data oriented and is hence closer to the graph structure of objects in OOP paradigm.

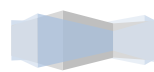
6. Bandwidth Performance
   - Due to metadata overhead, the same data takes more bandwidth if expressed in XML.
   - JSON data is highly compact using least amount of bandwidth.

These data exchange formats have their own advantages on one with other. The Table 1 and Table 2 show the advantage of each data exchange formats.

Summary

EAI is the combination of processes, software, standards, and hardware resulting in the integration of two or more enterprise systems allowing them to operate as seamless manner. To implement EAI used different integration architecture, ESB is the recent architecture to overcome the limitation of traditional architecture and realized the principle of SOA. SOA is an integration architecture approach that is based on the concept of a service. ESB provide different type of functions one of the most important for EAI is data transformation function. Data transformation function convert one data format and type to the other, its performances depend on data exchange format that implement in ESB and transformation function complexity level. There are different types of data exchange format but XML and JSON are mostly used data formats. Each format is more suitable for different situation JSON format is suitable for dynamic web applications and simple data transformation. It performance speed is faster than XML based format. XML based format is the most used technology for applying SOA because of easy to use and allows high interoperability.

SOA is used mostly web service communication. One of the main aims of Web services is to provide a loose coupling between service consumer and service providers. Web services transform XML documents into and out of IT systems. Web services use of several related XML-based technologies which are called: XML, WSDL, UDDI and SOAP.



3. Chapter Three: Related Works

3.1 Introduction

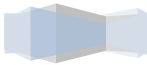
This chapter is reviewed a few research papers that related with the main concept of the research they are called ESB, data transformation performance, data transformation technologies and data exchange format. Mainly discussing implementation of SOA, ESB how to resolve different issues that appear in data transformation related with real time communication. The performances of data exchange format that related with widely used formats. Finally we will discuss an idea that related with this research and point out the limitation of the research.

3.2 SOA and ESB Implementation

Implementing SOA based software infrastructure which is ESB in a highly distributed, event-driven enterprise that resolve the issues which appear on traditional architectures, they are called hub and spoke and P2P integration. It allow regional control over the local integration domains; build choreography or business processes that can span departments or business units and remove the limitation that underlying MOM in its ability to cross physical network LAN segment boundaries and firewalls [14]. This paper presents an ESB-based EAI solution which is built on the basis of SOA. Through the ESB, the enterprise applications can achieve successful integration and the different heterogeneous characteristics of the application systems can be hidden. So the research conform that ESB is important due to cost effectiveness and efficiency for implementing EAI for enterprise and no matter how large or how small more it is suitable for any enterprise [14].

Traditional system architecture technology can solve the problem that appear in information shares and data exchanges inefficiently in power systems some extent but there are many Short comings, such as the closed system, strong dependence on vendors, tight coupling, poor reusability, poor expansibility, no uniform interface, etc. SOA combined with IEC61970/61968 standards provides a solution for the above problems. By analyzes the information system integration structure and the information exchange style among systems, finally propose a design scheme of ESB based on SOA [11].

The two pillars of IEC61970 are Common Information Model (CIM) and Component Interface Specification (CIS). CIM defines the semantics for contents of information exchange, while syntaxes for



information exchange are defined by CIS. IEC61968 is the specification of distributed system integration in power enterprises. A series of message types, which include the message definitions for basic classes of CIM, are defined to meet the requirement of CIM. Meanwhile, the power ESB meets IEC61970/61968 standards and integration of power information systems more flexible [11].

In solving the problem of heterogeneous systems in enterprise, this paper brings forward information exchange solutions of service-oriented architecture based on ESB centralized management to services, the scenario adopts services gateway interface design patterns and introduces efficient service scheduling algorithm, it achieves the decoupling between service packages and service communication, and it realizes efficient delivery of various information in the interactive process. It needs an interactive mechanism based on flexible, reliable and fast response to meet the needs of business development [26].

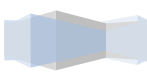
The design platform logic has effectively improved by using service gateway interface design patterns, it takes convenience to developers. In the platform design, scheduling algorithm is introduced and effectively solves the real-time of information exchange. ESB characteristics is fully used, it has overall design to platform and form a loosely coupled model, improve the overall system scalability and maintainability [26].

3.3 Data Transformation Logic

According to [10] most ESB platforms adopt XML-based model as their common data model, which is quite time consuming on data transformation. In addition to that the appearance of data transformation functions within single message is determine the complexity of the message that transfer from one system to the other through ESB, which affect real time communication performance.

The proposed solution for this issue is implementing Data Model for Integration (DIM) for integrating heterogeneous system in ESB. The method based on DIM to improve the efficiency of message transformation is by define the transformation rules for DIM, and give a common transformation algorithm which is implemented in Mediator of ESB framework for the data transformation. To create data model for integration and transformation algorithm [10]:

I. The Definition of Data Model for Integration
II. Model Transformation Rule based on:
III. The Transformation Algorithms Based on DIM



Finally the research conform that integration method based on DIM in ESB improves the processing performance when the quantity data transmission becomes large compared to the traditional method based on XML [10]. Proposing DIM is more stable for the company those have the process with the large size document. The limitation of this research is the experiments perform by considering different file size but not data transformation functions.

Even if XML Web Services is the most used technology for applying SOA, it suffer from a number of drawbacks such as low performance, bad utilization for hardware resources, and high network latency over remote procedure calls (RPC) implementation. These pitfalls could not be accepted under any circumstances in real-time communication in bank system. By conducting different experiments to mitigate this problem recommended the following tactics and techniques to overcome the problem [12].

- Utilize better encoders/decoders
- Leverage binary XML
- Apply data compression techniques
- Pre-generate serialization assemblies
- Divide large files
- Install silicon-based XML engines
- Apply parallelism techniques
- Utilize high speed networks

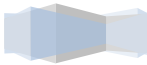
The limitation of the research is the study is focus on traditional integration methods which is RPC but not the resent ones which is ESB.

3.4 Data Exchange Format

Currently a lot of research was conduct to show the performance different between XML and JSON data exchange format by using different situation. According to [19] use those formats with different situation and recommended that XML and JSON are settable for the following scenarios:

XML can be used as a data exchange format in the following scenarios [19]:

- Performance of the application is not the primary concern
- Memory footprint of an application can be allowed to be large
- There needs to be strict adherence to a standard protocol for communication
- The data sent over the network is not of primary concern



On the contrary JSON can be used as a data exchange format in the following scenarios [19]:

- Performance of the application is a design consideration
- Memory footprint of an application needs to be kept minimal
- There can be deviation over the format of communication between parties or the parties agree to make necessary changes to the format as and when necessary
- The data sent over the network has to be optimized

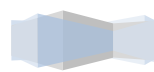
According to [27] even if XML and JSON serialization methods have similar function, they used in different application situations. These two forms of data can be use interchangeable XML serialization approach used for serialization data from third party web service to local web service. JSON is used with in local web service. This allows for better communication between web applications. By translating between each other, the data transmission of web applications can be secure, powerful in the XML serializing approach and fast and convenient via using the JSON serializing approach. In this way, the efficiency of web applications can be optimized, making applications more flexible and convenient [27].

Web applications will have better extensibility and adaptability in the aspect of data transmission allowing for transferring with different third party web applications. The use of translator layer structure can ensure that the translation works will have less impact on the speed of a web application, making this translation between XML and JSON more efficient. Also, such independent translator layer will improve the program structure of an application and improve the utilization of codes [27].

The efficiency of mapping data between different data models is the key point to improve the performance of web service applications. [26] This research was investigates how to employ JSON as the data exchange format for web service applications. Present an efficient approach to serialize and de-serialize JSON data in web service applications. Compared with XML, using JSON-style data for exchanging can improve the performance of web service applications. Their approach is based on the data mapping template generated by dynamic advanced binding technique through which the JSON data can be processed efficiently. Experimental results show that JSON performs better than XML in being parsed, being serialized and being desterilized [23].

Summary

This chapter is discussing on the papers which are related with the main concept of the research. Based on the papers ESB is the resent technology that improve the limitation that belong in traditional



architectures which are single point of failure and complexity. The performance of ESB data transformation is depending on data exchange format and data transformation complexity level. XML and JSON are widely used data exchange format. But JSON is more suitable for simple data exchange than XML format.

4. Chapter Four: The Proposed Technique

4.1 Introduction

This chapter discusses the implementation of ESB in Ethio telecom and proposed technique that used to improve the performance of data transformation. The chapter organized as follow: Section 4.2 ESB Implementation in ethio Telecom. Section 4.3 Proposed Techniques

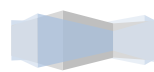
4.2 ESB Implementation in ethio Telecom

The company has greater than 50 million customers and used different systems to manage their requests and make the service available from them. To manage these activities, it uses different systems and work in seamless manner. To integrate and overcome the drawback of transitional integration architected, improve interoperability and business activity, it was implemented the recent integration technology which is ESB during the implementation of telecom expansion project (TEP1). ESB is used to communicate the systems those required real time communication. The BSS solution and ESB was implemented by the same vender at the same time. The third party system was implemented with different time and venders with BSS systems.

The implementation is creating interoperability with in heterogeneous systems in the company and across the company. In the company ESB is used to create the communication which is within BSS and BSS with third party systems for real time communication. It helps to transporters to construct a service-oriented ESB and IT infrastructure platform more quickly, flexibly, and efficiently [29]. In addition, it aids to communicate with different applications between external companies such as Commercial Bank of Ethiopia and ethio telecom. ESB is providing different functions for integrate different systems, some of them which activate in the company are:

- Message Routing
- Message Mapping
- Message Forwarding
- Message Transformation

Majority of business processes use this technology to complete the task with in different systems, some of systems are that handle real time processes are CRM and CBS. Those systems are communicated through ESB and used to manage company day to day business activity. So any issues that are related



with ESB data transformation performance is directly related with the company business activity and service management. The Figure 4.1 is show as the communication among of BSS and third party systems with ESB solution [29].

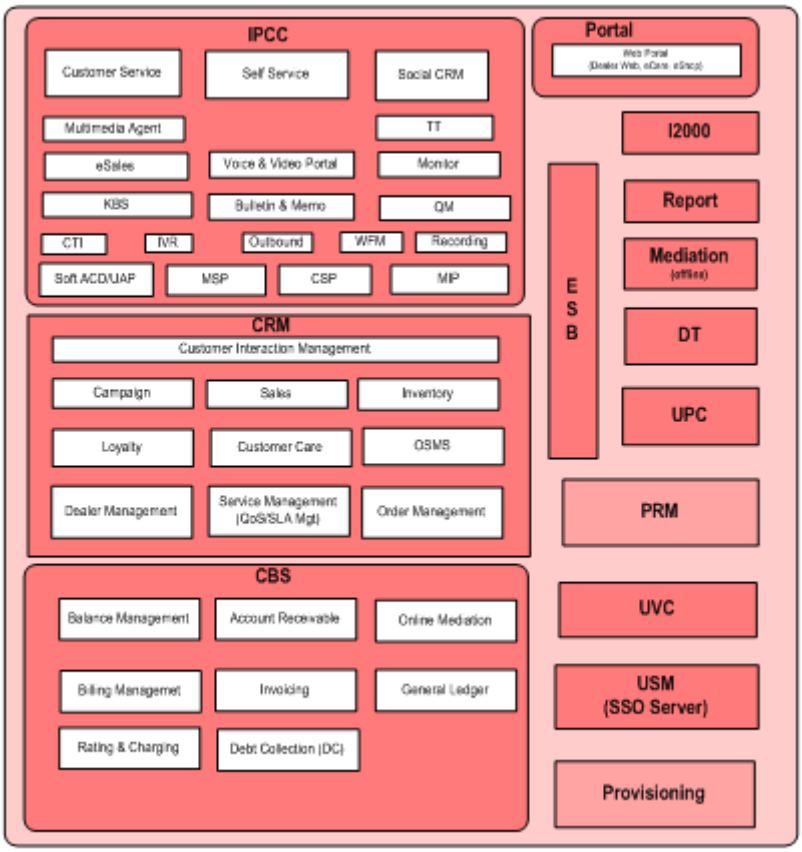


Fig 4:1 General view of BSS, Third Party Systems with ESB [30]

4.2.1 ESB Interface

Figure4.2 is shows us the interface with in ESB in ethio telecom it has an interface from BSS solution and third party systems. As we can see from the diagram the interconnection with inter organization and inter organization systems. For example Bank system is the external system, CBS, CRM and IPCC are from BSS solution but the remaining systems are from third party. Except Bank's system all are within inter organization. This research is selecting the scenarios to laboratory experiment from these connections like the communication among Bank and CBS and CRM and CBS.

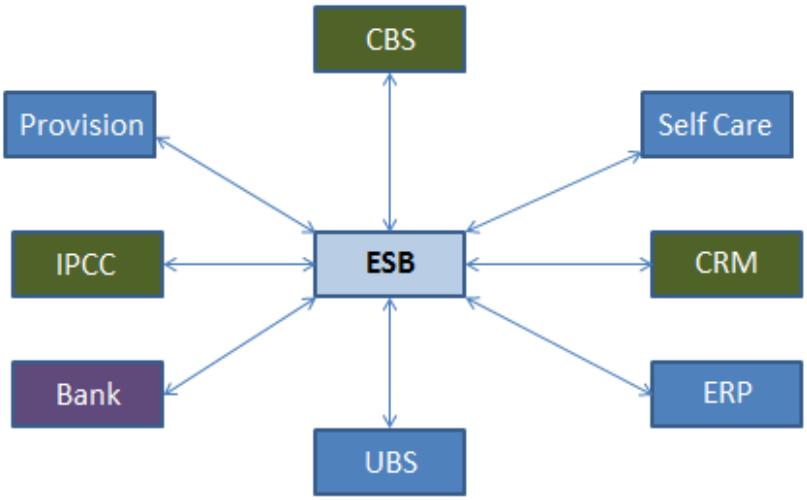


Fig 4:2 ESB Interface [31]

4.2.2 Real-Time Access Information Collection

On ESB component always records the message of the request and response to DB regardless the business success or failure. It's easy for maintainer to track the log based on message level. The 'Error Agent', 'Info Agent' and 'Debug Agent' in the Figure 4.3 diagram are the fundamentals for the error handling [29].

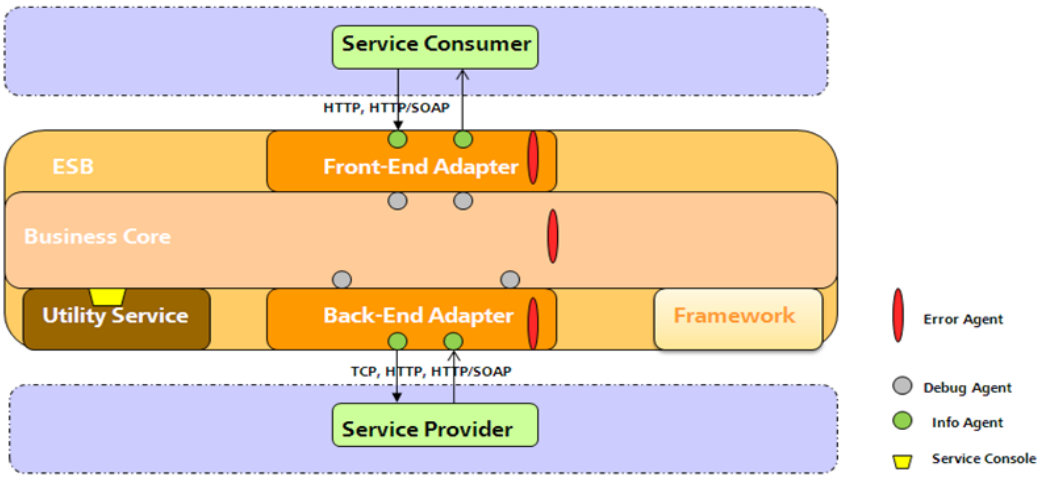


Fig 4:3 Enterprise Service Bus Architecture [29]

In this diagram Front-End Adapter is used to transform the data that come from source system data model to common data model. On the other side Back-End Adapter is used for transform the data from target system data model to common data model. The middle part is representing the full function of ESB one of them are data transformation.

4.2.3 Data Model Transformation

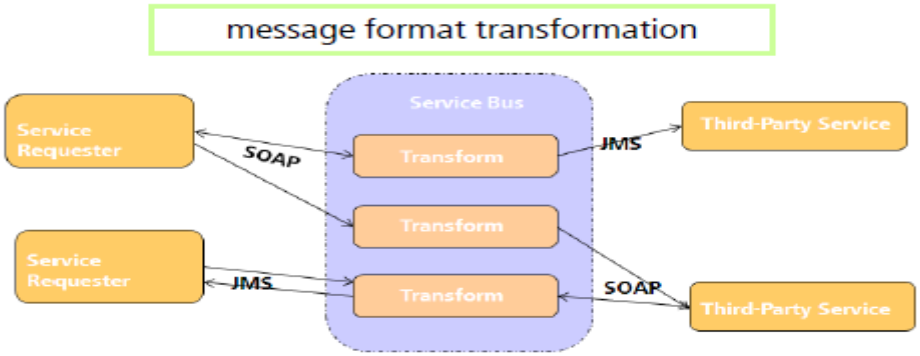


Fig 4:4 ESB Message Format Transformation [32]

Simple Object Access Protocol (SOAP)**:** Based on W3C specification SOAP is an application communication protocol, it is a format for sending and receiving messages, and platform independent. In addition to this SOAP work based on XML format [32].

Java Message Service (JMS): it was initially developed to provide a standard Java API for the established messaging products that already existed. JMS provides a common way for both Java client applications and Java middle-tier services to use these messaging products. It defines some messaging semantics and a corresponding set of Java.

4.2.4 Message Conversion and Routing

Message conversion and routing are some of ESB functionalities which are crucial for system integration. Figure 4.5 shows as how to perform data transformation and routing by using ESB in the company.

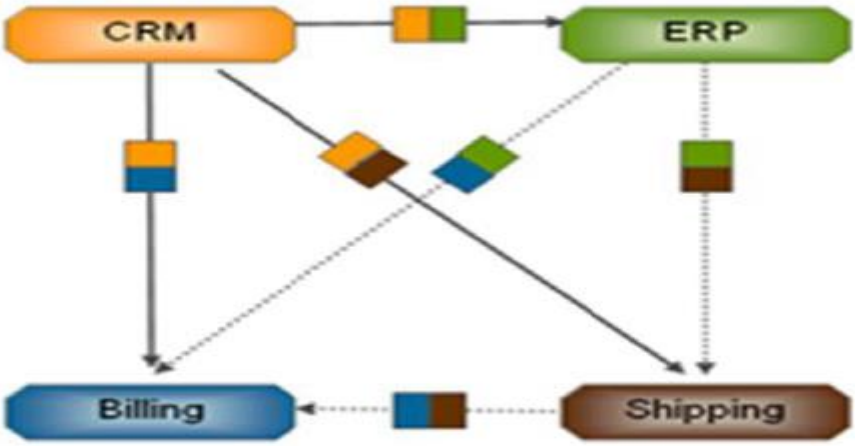


Fig 4:5 Message Conversion and Routing [31]

4.2.5 Integration Implementation View

The ETH Switch system integrates with CBS via payment interface with SOAP protocol. Airtime Recharge service is one of the real time processes between external system which is ETH Switch system (which is commercial Bank of Ethiopia system) and ET system The following diagram illustrates the integration structure of Airtime Recharge service between ETH Switch system and ET CBS [29].

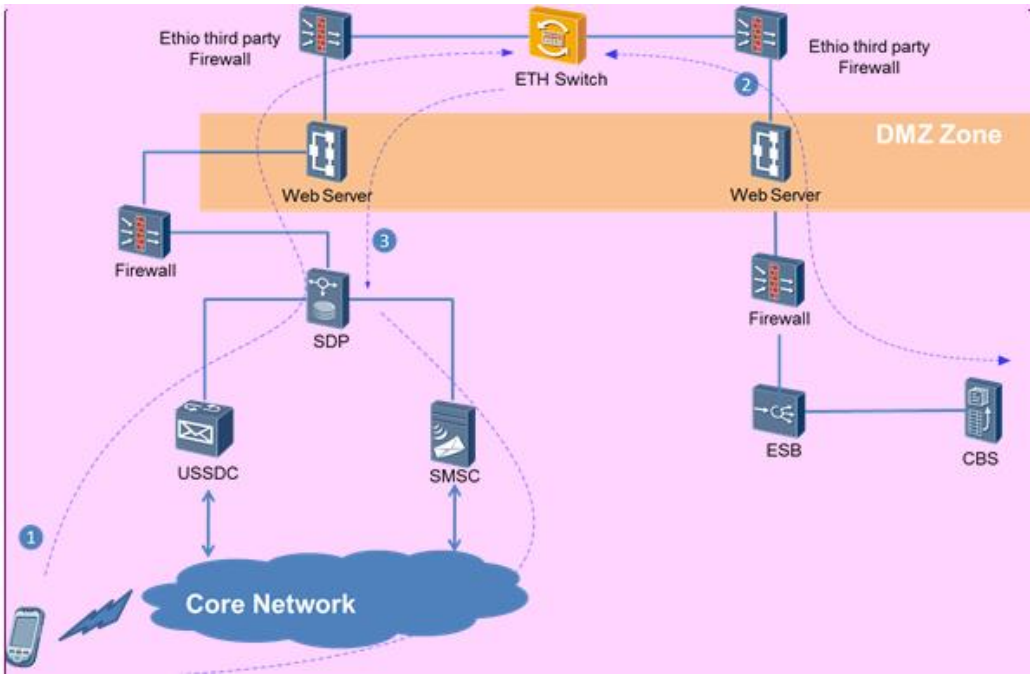


Fig 4:6 Airtime Service Architecture View [29]

1) Customer can initiate airtime recharge request by handset, and request will send to ETH Switch system through USSDC and SDP.
2) ETH Switch system will send the recharge request to CBS system through SOAP protocol after validate customer's request.
3) ETH Switch system will send notification to Customer through SDP and SMSC after get the response from CBS.
4) The connection between ET swatch MM system and ET CBS is through IP SEC VPN.

4.2.6 The Main Flow of Airtime Recharge Service Flow

The below real time business process is one of real time process that occurs between ethio telecom system's and external customer system's which is Bank system's [29].The following diagram illustrates the service flow of Airtime Recharge service:

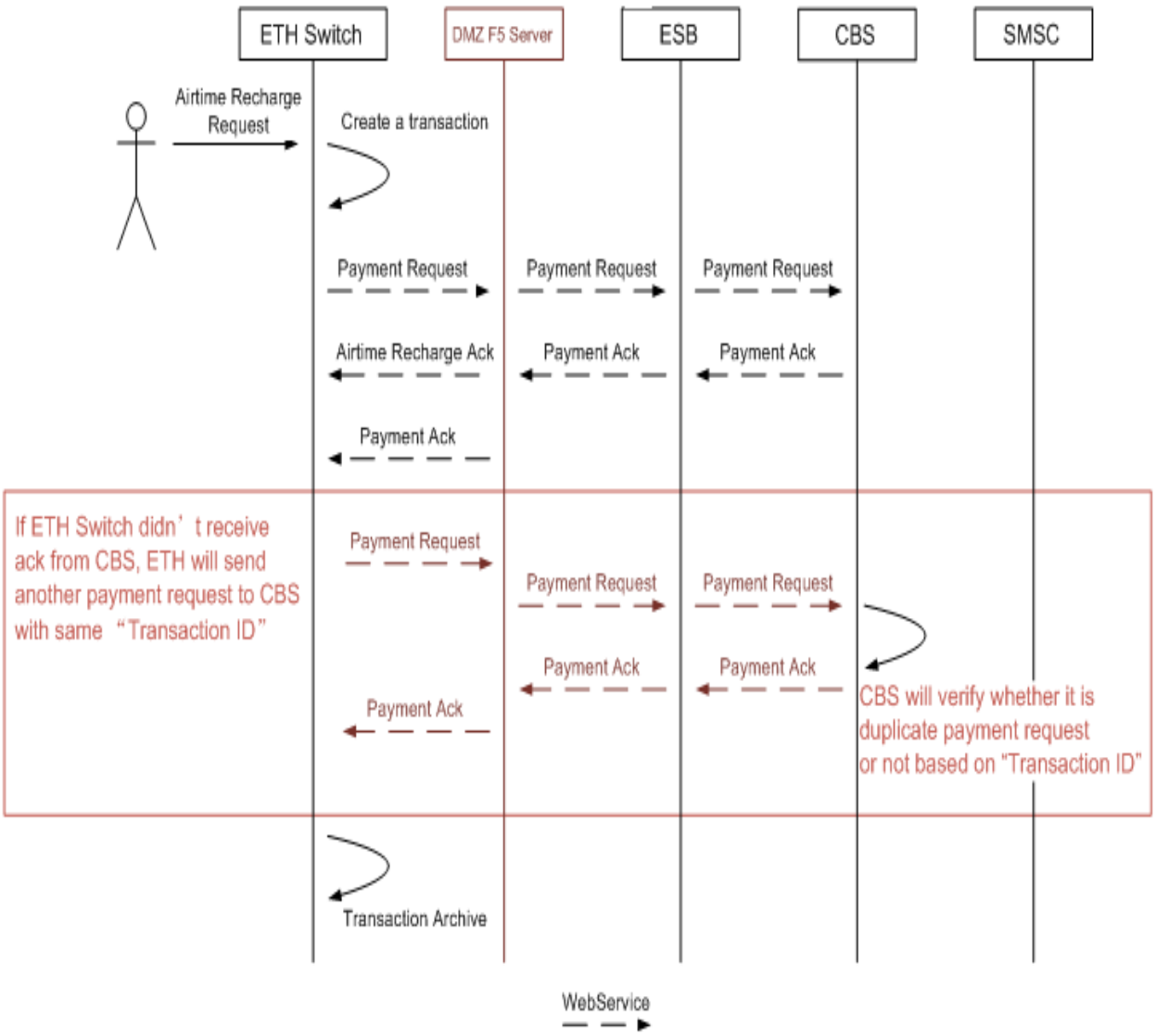


Fig 4:7 Service Flow for Airtime Recharge Service

1) Customer initiate "Airtime Recharge" request
2) The request will be forward to ETH Switch system
3) ETH Switch system will validate the transaction according to the specific business rule.
4) On successfully validating these rules, the transaction will become to the 'Authorized' status at this moment.
5) Then, the ETH Switch system will send an external "Payment" request to DMZ F5 Server in
6) Real time via SOAP API

DMZ F5 Server will do security authorization and forward payment request to CBS through ESB, and wait the response from the ESB.

7) CBS will do payment operation internally, and response result to ETH Switch system through ESB/DMZ F5
8) If the correct response is received from the CBS, the ETH Switch system will complete the corresponding transaction. The transaction status will be changed to 'Completed'.
9) After the transaction is completed.
10) If ETH Switch system can't get any payment response from BSS (ESB) within specific period, ETH Switch will send payment request again with same "Payment Serial No". And BSS should response result to ETH Switch as below logic:

    a).If CBS didn't process the previous payment request, CBS will continue the payment process internally, and response result to ESB, ESB should forward response message to ETH Switch.

    b).If CBS already did payment process in previous request, CBS should not accept the payment amount, and response successful result to ESB, ESB should forward response message to ETH Switch. As well if error happens error code returned to ETH Switch.

11) If ETH Switch system failed to get the response within 3 times, ETH Switch system will leave the transaction in "Authorized" status, and require manually operation or reconciliation process.

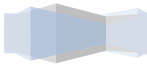
Note: For Prepaid subscriber, CBS will add money to its main balance account based on "Payment amount".

4.3 Proposed Technique

4.3.1 Data Complexity Matrix Level

Complexity level is categorizations which represent the time required for doing data transformation based on complexity matrix. Data complexity matrix leveling and complexity analysis algorithm are the key factors for proposed solution.

In my knowledge, I couldn't get any indicator that used for data transformation complexity matrix leveling. So to set that level this research performs different kinds of experiments. According to [10] while the data transformation function increase the complexity level also increase in the same manner,



Due to this reasons, data transformation functions combination experiment result select only two of them from four different data transformation functions. The time required for doing three and four possible combinations of functions have significant different as compare with single and two possible combinations of functions. Finally select maximum value from single transformation function and minimum value form two possible transformations function and take the average of those values for complexity matrixes leveling.

By considering those idea complexity matrix is developed based on the results of Section 5.4 and Section 5.5. When transformation is done for individual functions, the highest required time is 32ms for *Value Transformation*. In the other hand, minimum value for two possible functions combination is 45ms which is the result of *Name transformation and attribute aggregation*. So the average value of those results is around 39ms. Then the complexity analysis algorithm used this average value to check the complexity level of the message. For complexity level is less than 39ms will be pass XML format as it is without any conversation. But, when complexity level is greater or equal to 39ms will be convert in to JSON file.

Based on the average value complexity matrix categorized into two parts as follow:

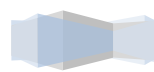
1. The time required for doing individual or single function transformation are less than 39ms.
    - The times require < 39ms => Low Complexity Level (LCL)
2. The time required for doing more than one functions transformation are greater than 39ms
    - The times require > =39ms => High Complexity Level (HCL)
    - This value indicated that the time required for doing data transformation functions by used any possible combinations but not considering the single function.

Table 3 Complexity Matrix

Data Transformation Type	Complexity Level
1, or 2 or 3 or 4	**Minimum Complexity Level (MinCL)**
1 &2 or 1 & 3 or1 & 4 or 2 & 3 or2 & 4 or 3 & 4 or1,2 & 3 or1, 3, & 4 or 1,2 & 4 or 2,3, &4 or1,2,3 &4	**Maximum Complexity Level (MaxCL)**

Note: the time require for each case are list in section 5.4 laboratory experiment results.

- MinCL =the message that content single data transformation function only.



- MaxCL =the message that content two, three or four possible data transformation function combination**.**

4.3.2 Complex Analyses Algorithm

The main purpose of this algorithm is to check the message content by using complexity matrix and determined the complexity level (high or low) and based on the result data exchange format will assign. Figure4.7 shows the algorithm procedure.

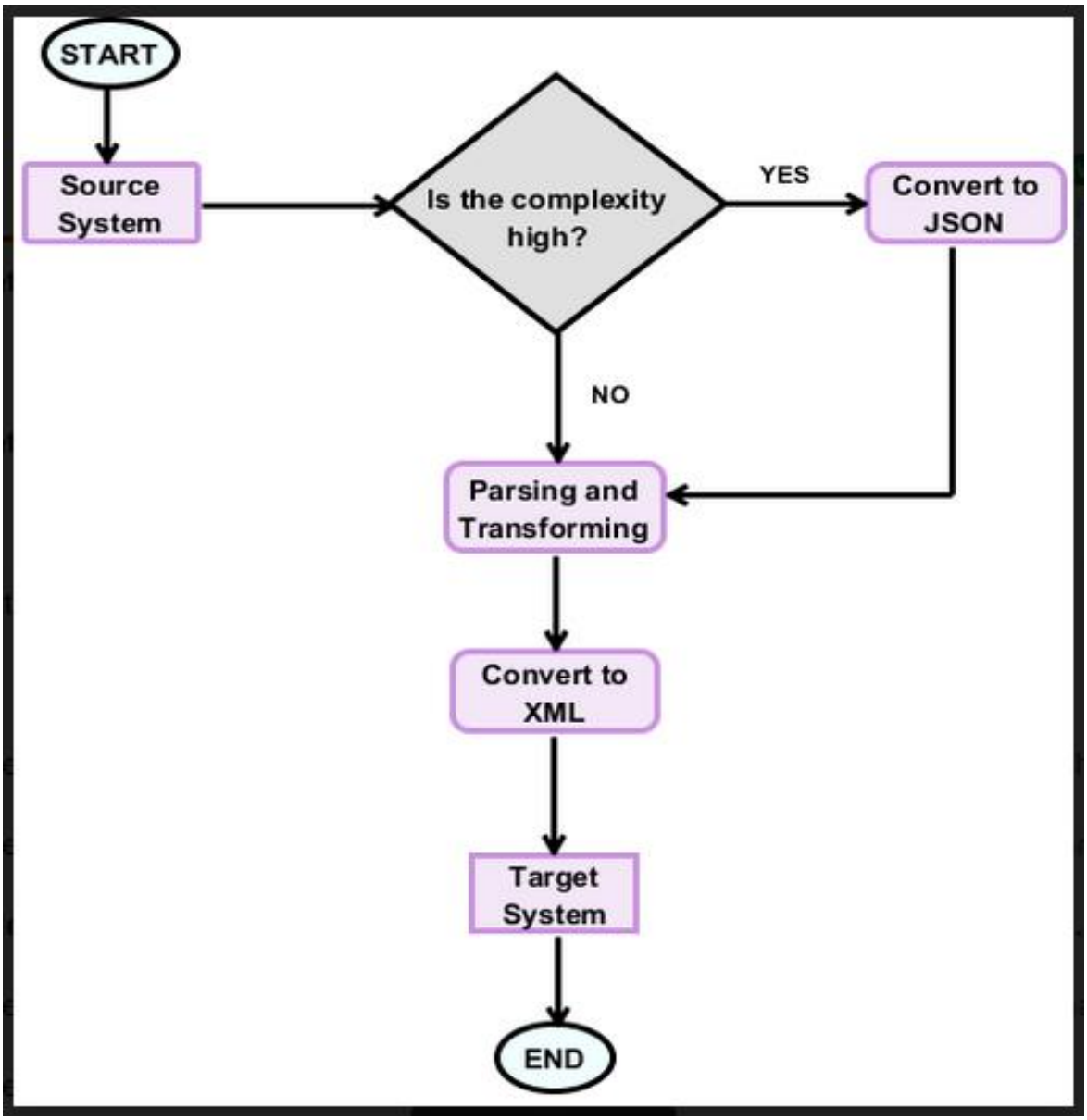


Fig 4:8 Complexity Analyses Algorithm Process Flow

4.4 The Propose Technique

For data transformation efficiency problem, this research proposes *an improved technique for enterprise service bus data transformation*. The technique is use XML and JSON data exchange formats based on the complexity matrix level. To select one of the data format based on complexity level the solution use complexity analyses algorithm. The proposed solution uses different components which are called: ESB, CAA, source and target systems. Figure 4.7 illustrate the proposed solution data flow, the place of CAA and data transformation.

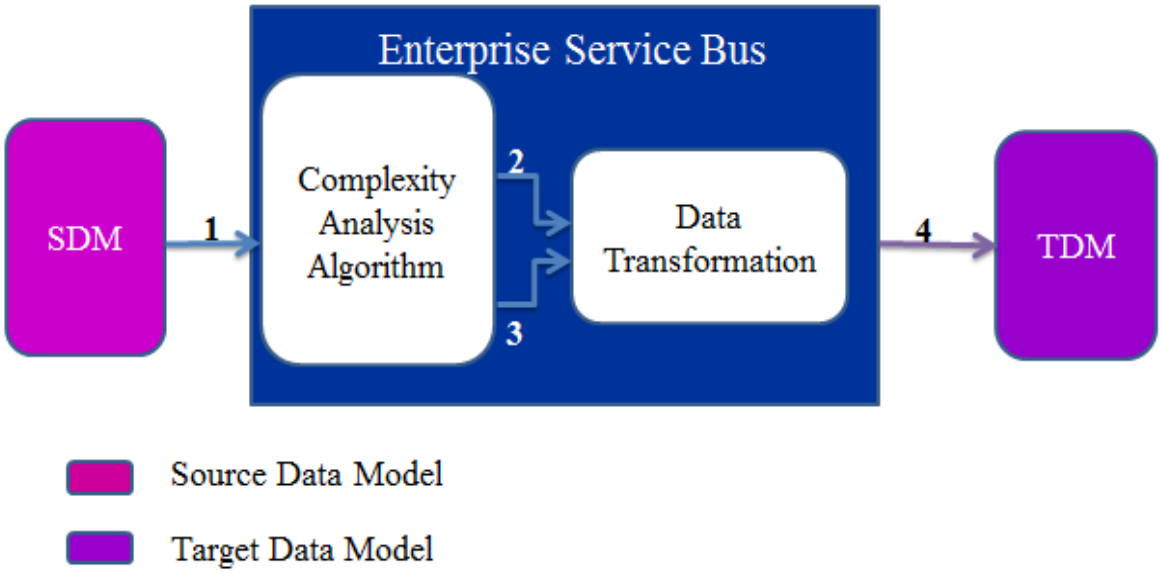


Fig 4:9 Proposed Solution for Improve the Performance of Data Transformation.

4.4.1 Number and Component description

- "1" contents the data that come from source systems in the form of XML data exchange format.
- "2" the output of complex analyses algorithm (CAA): that indicated the data with low complexity level in the form of XML format.
- "3" the output of CAA: that indicated the data with high complexity level. At this part XML format should convert to JSON format, so this number represent the data with JSON format.
- "4" after doing data transformation the message convert to XML format and send to the target systems. This task is not including in the research evaluation part.

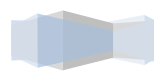
Enterprise Service Bus (ESB): is the middleware to integrate heterogynous systems to work in seamless manner. One of the main ESB function for integration is doing data transformation based on the mapping source and target systems data formats. This task mainly used the functions those are list in Section 4.5. The main task of the proposed solution is to receive message from the source systems, parsing and then mapping with target system's data format and type. Based on mapping result perform data transformation then convert to XML format finally pass the message to the target system. Source Systems: In order to communicate two heterogynous systems message request should be initiates from source systems by its own data format in the form of XML based data exchange format**.** So **s**ource system send request message to target system through ESB.

Complexity analyzer algorithm (CAA): The main objective of this part is analysis the mapping results and complexity level then selected data format to exchange the data. If the data with low complexity level pass without any conversion, otherwise convert to JSON format. The algorithm is mention in Section 4.6. In general the output of CAA are in the form of XML or JSON data exchange formats based on complexity level and forward to data transformation part for parsing and data transformation.

XML and JSON data exchange format: these part are used for data exchange based on complexity level.

Target System: It is service provider and response the request through ESB during message respond time. At this time this part represent as source system and the source system become the target system.

Initially message request come from the source system in the form of XML format. Then pass to CAA the main task of complexity analyses algorithm is check the complexity level, based on the complexity matrix level then used one of the exchange formats. For the data with low complexity level (<39ms) use XML format or pass the incoming message as it is, but for high complexity level (>=39ms) by converting XML format to JSON format to exchange the data. These complexity levels are determined by the number of data transformation functions that appears with in a single message. When the message passes to data transformation part in the form of XML or JSON format; first it performs parsing the incoming data based on data format then message transformation based on demand. After that the message converts in to XML format and forward to target system, this process is used for message request flow in ESB.



Summary

IBM web sphere ESB was implemented in ethio telecom for communicate BSS systems and BSS with third party systems for real time communication. It used XML based data exchange model to move the data with in different applications, but it has performance issue for real time communication. This research has been proposed an improved technique for ESB data transformation. The proposed solution is content two different data exchange format which are XML and JSON format based on data complexity level. These complexity levels are determined by complexity matrix and applied in complexity analyses algorithm. The proposed solution is used XML based data model for the message with low complexity level but JSON format is used for the data with high complexity level.

5. Chapter Five: Evaluation

5.1 Introduction

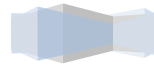
Many researches have been done to show the performance difference between XML and JSON data exchange formats and conform that performance of JSON has better than XML. [19] As far as performance concerned, the JSON data exchange format is the clear winner from XML format in terms of both memory footprint as well as the parsing runtime, JSON delivers better performance at the cost of readability and flexibility. But this evaluation is different form the other; so doing an experiment to show the performance difference between those two formats related with ESB data transformation based on data transformation functions complexity level. The chapter is organized as follows: Section 5.2 System Set up and Evaluation Description Section 5.3 Re-Experimenting. Section 5.4 Proposed Solution Evaluations. Section 5.5 Experiment Results and Discussion.

5.2 System Set up and Evaluation Description

In order to evaluate the main objective of this research, two main evaluations are done. The first one is Re-experiment which is use to set complexity matrix level for data transformation by the time required to doing possible combination of data transformation functions. The second one is used to check the proposed technique achieve the objective of the research or not.

Simple API for XML (SAX) model is use to parsing XML format, Jackson is use to parser JSON and Java programing language. Lap-top with the following specification: Intel® Core(TM) i7 - 4610M CPU @ 3GHz, 4.00GB (2.92 GB usable) memory (RAM), 32 bit Operating system, Java SE Development Kit 8 update 171, Net beans IDE 8.2 , rating 4.7 windows experience Index and Windows 7 Ultimate. The reason to select SAX is, an event-based streaming model typically used to process large XML documents without actually building a parse tree [33]. SAX parsing doesn't require more space and time compare to other model, which is Document Object Model (DOM).

1. Re-Experimenting: The purpose of this experiment is to find out the time required for performed data transformation and message parsing independently and jointly. By using ethio telecom as the case study, three different scenarios are selected. This experiment demonstrate



transformation for a single function and for different possible combination function, specifically the experiment used the combination that mention in complexity matrix (Table 3).

2. Evaluation of Proposed Technique: this experiment is used to show the end to end evaluation of the proposed technique. It is use XML and JSON data exchange format independently based on complexity matrix leveling and complexity analysis algorithm, those detail description are mention in Section 4.5 and 4.6.

5.3 Re-Experimenting

This experiment is used to find out the time required for parsing by using selected scenarios in the form of XML and JSON data exchange formats and data transformation.

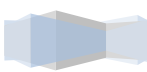
1. Parsing JSON and XML file formats: For doing this experiment first select three different scenarios from ethio telecom which required real time communication. One of the selected scenarios is indicating that communicate between *Commercial Bank system* and *ethio telecom system (CBS)*. The reaming two scenarios are indicating that the communication between two systems with in ethio telecom, those systems are *CRM* and *CBS*.

   Scenario 1 (S1): The first experiment is indicating that the time requires to parsing *Air Time Recharge Request Message*. This message is sent from Commercial Bank of Ethiopia (CBE) to ethio telecom or Request Message from ETH Switch system (CBE) to ET CBS system and Response Message from CBS to ETH Switch.

   Scenario 2 (S2): The second experiment is the Message Request from CRM to CBS which is called *Change Offering during Reactivation* and Respond Message from CBS to CRM. Both systems are belonging to BSS solution in ethio telecom.

   Scenario 3 (S3): The third experiment is that the Message Request that create during the process of *Change SIM Card during Reactivation*. Message Request initiate from CRM to CBS, and Respond Message from CBS to CRM.

2. The second experiment is indicated that find out the time required for performing different type of data transformation functions and parsing. Four experiments are done for this case:
   1. Find out the time require for performing one of the data transformation functions at a time.



2. Find out the time require for performing any combination of two different data transformation functions.
3. Find out the time require for performing any combination of three different data transformation functions.
4. Find out the time require for performing any combination of four different data transformation functions.

5.3.1 Experiment One

Scenario 1**:** Find out the time required for parsing three selected scenarios by using XML and JSON *format.*

Note**:** In all experiment the time is measured in millisecond.

Table 4 shows that the time required for parsing the content of Message Request and Message Response of those selected scenarios by using XML and JSON data exchanging format. From those scenarios S1 and S3 message content are almost the same.

Table 4 Parsing Results of the Selected Scenarios

Scenarios	**XML Execution Time Usage (ms)**	**JSON Execution Time Usage (ms)**
Change Offering During Reactivation Recharge Request Message S1	171	47
Change Offering During Reactivation Recharge Response Message S1	47	15
Air Time Recharge Request Message S2	78	31
Air time Recharge Respond MessageS2	30	15
Change SIM Card during Reactivation Request Message S3	78	46
Change SIM Card During Reactivation Response Message S3	47	15

Discussion

Discuss the result of this experiment in three different points of view

1. Parsing result of XML format: XML format parsing result indicated that, the time required for parsing is directly proportional to number of object.
2. Parsing result of JSON format: similar with XML format while parsing JSON format the time requires for parsing increase while the number of object increase. But the incremental amount is less than the incremental amount that required in XML format.
3. Parsing result of XML and JSON formats: while compare those result with each other, JSON format is require less time than XML format for parsing based on the same number of object of the data.

5.3.2 Experiment Two

1. This experiment indicated that the time required for doing data transformation by using each data transformation functions. Table 5 indicated that the result for doing single data transformation function independently.

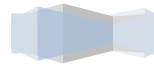
Table 5 The Time Require for Doing Each Data Transformation Functions

Parameters Types	**Execution Time in (ms)**
Object's name transformation	16
Object's value transformation:	32
Attribute and Value Aggregation	15
Attribute and Value Splitting	16

Discussion

2. This experiment indicated that the time requires for doing *Value Transformation* is required more time compare with other functions. But the remaining transformation functions are required almost similar time.

Note: For the reaming experiment used experiment two result as a reference.



3. This experiment indicated that the time required for doing two possible combinations of transformation functions. Table 6. Shows as the result of each possible combination

Table 6 The Time Required for Doing Two Possible Combination of Data Transformation

Parameters Types	Execution Time in (ms)
Name and value transformation	52
Name transformation and attribute aggregation	45
Name transformation and attribute splitting	62
Value transformation and attribute aggregation	67
Value transformation and attribute splitting	53
Attribute aggregation and splitting	51

Discussion

- The result indicated that the time required for *Name transformation and attribute splitting* and *Value transformation and attribute aggregation* required more time to compare with the other results.

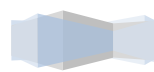
4. This experiment result indicated that the time required for doing three possible combinations of functions. Table 7 Shows as the results of each possible combination.

Table 7 The Time Required for Doing Three Possible Combination of Data Transformation

Data Transformation Types	Execution Time in (ms)
Name and value transformation and attribute aggregation	78
Value transformation and attribute aggregation and splitting	80
Name transformation, attribute aggregation and splitting	55

Discussion

- The result of this experiment indicated that when the data complexity increase the time require to doing that tasks also increase and the transition time which is moving from one task to the other also increase.



5. This experiment indicated that the time required for doing four possible combinations of functions. Table 8 Shows as the results of each possible combination.

Table 8 The Time Required for Doing Four Possible Combination of Data Transformation

Data Transformation Types	**Execution Time in (ms)**
Name and value transformation, attribute aggregation and splitting	**93**

Discussion

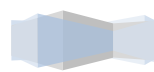
- The result of this experiment indicated that similar with the experiment III when the data complexity increase the time require to doing that tasks also increase and the transition time which is moving from one task to the other also increase.

5.3.3 Experiment Three

- In this experiment find out the time required to parsing and transform using four different data transformation functions. Used one of the selected scenarios which are *Air Time Recharge Request Message* that is sent from Commercial Bank of Ethiopia (CBE) system to ethio telecom system. The experiment used XML and JSON data exchange formats, Table 9 shows the results of parsing and data transformation together.

Table 9 The results of Parsing and Data Transformation by XML and JSON Formats

Combination of Parsing and Data Transformation	**As-Is XML Execution Time in (ms)**	**As-Is JSON Execution Time in (ms)**
Parsing with Name Transformation	94ms	78ms
Parsing with Name and Value Transformation	141ms	93ms
Parsing with Name and Value Transformation and Attribute Aggregation	172ms	110ms
Parsing with Name and Value Transformation, Attribute Aggregation and splitting	203ms	141ms



Discussion

The results of the experiment are discussed by grouping in to three parts:

- Compare the results of XML data exchange format: The results indicated that while the complexity increases the time required for doing the task also increase. But the increment amount within XML results are greater than the values that appear in JSON format.
- Compare the results of JSON data exchange format: Similar with XML format, while data transformation complexity increases the time required is also increases. But the increment amount of time for JSON format is less than that of XML format.
- The result comparison between XML and JSON data exchange formats by increasing complexity level: While we compare the required time between XML and JSON, the time required for parsing and data transformation by using JSON format is less than XML data exchange format, which indicated that the JSON format is suitable for time sensitive processes compare with XML format.

5.4 Proposed Solution Evaluation

This experiment is mainly performing two different evaluation which are functional testing and End-to-End (E2E) performance testing for proposed technique. For the functional testing first we grouped the message type in to two parts which are low and high complexity level. The time less than 39ms takes as low complexity level and time greater than or equals to 39ms is for high complexity level; the detail description are state in Table 3.

For E2E performance testing, the evaluation used four different cases, they description as below:

- Case One: This scenario is use single data transformation function which indicated that complexity matrix with low complexity level. Parsing and Name transformation from XML data format. Here we can use any one of data transformation function at a time.
- Case two, three, and four: These cases are indicated that performing different combination of data transformation functions at a time. It could be two or three or four functions appear with in a single message that means the message with high complexity level. At this time convert the incoming message from XML format to JSON format then parsing and data transformation perform on JSON format. For these cases used complexity matrix leveling from Table 3.

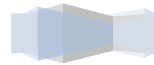


Table 10 Evaluation Result for To-Be Data Exchange Model

Evaluation Cases	To-Be XML/JSON Time
One	109ms
Two	125ms
Three	141ms
Four	187ms

From Table 10 indicated that similarly with the previous evaluation while the complexity increases the time require also increase. Data transformation complexity is determined by the appearance of transformation functions within single message.

5.5 Experiment Results and Discussion

For doing functional testing the program is execute based on the time, when the time is less than 39ms the output indicates that message with low complexity level, so pass the message as it is. But if the time is greater or equal to 39ms the output indicated that the message with high complexity level so convert the incoming data which is XML format to JSON format based on the complexity analysis algorithm. From this functional testing result we can conclude that the functional testing is successful without any limitation.

In other way when we compare within As-Is results which means the result of parsing and data transformation without complexity matrix and CAA by using XML and JSON format. The time require for JSON format is less than XML format which indicated that at high complexity level that mention in Table 5.7 last scenarios JSON format improved the performance of XML format by 31% at high complexity level or for the case of doing *Parsing with Name and Value Transformation, Attribute Aggregation and splitting.*

According to [11] JSON and XML provide unique strengths, but the importance of performance and resource utilization must be understood when making decisions between data interchange formats. In terms of both memory footprint as well as the parsing runtime, JSON delivers better performance at the cost of readability and flexibility [19]. In similar manner in this research we can confirm that the performance of JSON format has better performance than XML in parsing run time value.

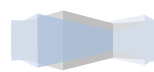
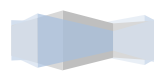


Table 11 The Time Require for As-Is and To-Be Format by Using XML and JSON Formats

Combination of Parsing and Data Transformation	***As-Is XML Execution Time in (ms)***	***As-Is JSON Execution Time in (ms)***	***T0-Be XML/JSON Execution Time in (ms)***
Parsing with Name Transformation	**94ms**	**78ms**	**109ms**
Parsing with Name and Value Transformation	**141ms**	**93ms**	**125ms**
Parsing with Name and Value Transformation and Attribute Aggregation	**172ms**	**110ms**	**141ms**
Parsing with Name and Value Transformation, Attribute Aggregation and splitting	**203ms**	**141ms**	**187ms**

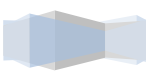
When we come to E2E performance testing or proposed solution evaluation, for the first case the result of To-Be is greater than the result of As-Is. Because, the result of To-Be required additional time for doing complexity analysis compare with As-Is. On the reaming three scenarios the incoming data format should be converting to JSON format. So the To-Be results indicated that the proposed solution improved the performance of XML format as shown in Table 11. So in this specific case the proposed solution is improved the performance based on complexity level.

The results that indicated for the case two improved by 12%, for case three improved 18%, but for case four improved 8%. Even if the results show that in the last case improvement is less than compare with the others cases. In my opinion the possible case for this happening could be JSON format has lack of throughput performance, the other one is the impact of data splitting in XML format is less compare with JSON format. In the case of ethio telecom ESB is not used for transferring large amount of data. So those both assumptions will require further study.



Summary

This chapter is mainly focus in the evaluation and discussion point of the proposed solution and re experimenting by considering ethio telecom as case study. The result indicated that the performance of data transformation is depending on the complexity level of the message. And JSON format is suitable for time sensitive processes.



6. Chapter Six: Conclusion, Recommendation and Future Work

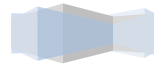
6.1 Conclusion

As far as performance is concerned many research has been done and show performance difference between XML and JSON data exchange format. This research also shows that ESB data transformation performance difference between those formats by considering ethio telecom as a case study. First this research analyze the performance of implemented ESB specially in data transform function and proposed an improved technique for enterprise service bus data transformation. This technique is use XML and JSON data exchange formats based on data complexity level independently. XML based format is used for the data with low complexity level and JSON format is used for high complexity level.

The proposed technique is used two main concepts which are complexity matrix and complexity analysis algorithm. Complexity matrix is used to categorize the complexity level in to two parts which are low and high complexity level. Complexity analyses algorithm is used for select data exchange format based on complexity level that maintain in complexity matrix. The algorithm first check the complexity level and decide which data exchange format should use. While the complexity level of the data is high it convert the incoming message to JSON format and facilitate for parsing and data transformation otherwise pass the incoming data as it is. The proposed solution evaluation result indicated that, the proposed technique is improved the perforce of ESB data transformation. And JSON format is suitable for real time communication compare with XML format at high complex level.

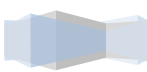
6.2 Recommendation

In the company large number of subscribers request and business process transaction are processed by using real time communication. The performance of real time communication is the key factor to determine the managements of those activities. So by using the proposed solution the company can minimized the issues which related with real time communication and maximize its revenue. In addition to this the company can maximize customer satisfaction and minimize human intervention to solve the issue that reacted with this specific area.



6.3 Future Work

For the future work, this research recommends to test the impact of XML and JSON data formats related with other ESB functions related with the performance of real time communication.



References

[1] L. D. Xu, "Enterprise Systems: State-of-the-Art and Future Trends," *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS,* Vols. VOL. 7, NO. 4, NOVEMBER 2011.

[2] T. . R. Soomro and A. . H. Awan, "Challenges and Future of Enterprise Application Integration," *International Journal of Computer Applications (0975 – 8887),* vol. Volume 42– No.7, March 2012.

[3] S. L. David , Enterprise Application Integration, First Edition November 05, 1999.

[4] H. Jörgen , O. Jens , . E. Nevzat and J. Paul , "P2PIE: A New Enterprise Application Integration Solution," 2015.

[5] . R. Mohammad and A. A. Goknur , "Enterprise Application Integration (EAI), Service Oriented Architectures (SOA) and their relevance to e-Oriented Architectures (SOA) and their relevance to esupply," *African Journal of Business Management,* Vols. Vol. 4(13), pp. 2604-2614, 4 October, 2010.

[6] K. Kabul and A. Ahmad , "Service Orchestration using Enterprise Service Bus for Real-time Government Executive Dashboard System," in *2015 International Conference on Data and Software Engineering*.

[7] D. Pingshun, "Design and Implementation of ESB Based on SOA in Power System," 2011.

[8] . P. A. Sanjay and P. Amit , "Enterprise Service Bus: A Performance Evaluation," *Communications and Network, 2011, 3, 133-140.*

[9] L.-m. SHI and G. WANG , "The analysis of a data transformation method based on ESB," 2011.

[10] . F. Jing and X. Cong , "Data Transformation of ESB Based on the Data Model for Integration," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012)*.

[11] . N. Nurzhan, P. Michael and R. Randall , "Comparison of JSON and XML Data Interchange Formats: A Case Study," 2009.

[12] . M. R. Alaa, E. H. Ahmed and F. H. Qusay , "Investigating Performance of XML Web Services in Real-Time Business Systems," *Journal of Computer Science & Systems Biology - Open Access,* vol. JCSB/Vol.2, September-October 2009.

[13] G. Anjali , "Enterprise Application Integration," *SPAN White Paper,* 2014.

[14] W. Jieming and T. Xiaoli , "Research of Enterprise Application Integration Based-on ESB," 2010.

[15] . K. Martin, A. Oscar, H. Sarah , . H. Andrew, . K. Hanumanth and . N. Alasdair, Patterns: SOA with an Enterprise Service Bus in Web Sphere Application Server V6, May 2005.

[16] K. Martin , B. Thomas, . P. D. Aditya, . F. Bernardo, H. Andrew, L. Nay , O. Fatima i and S. V. Jesús Ángel , Smart SOA Solutions with Web Sphere Enterprise Service Bus Registry Editiion V7.5, 2011.

[17] N. Eric, Understanding Web Services XML.WSDL, SOAP, and UDDI, 2005.

[18] A. Tommi and P. Tuomas, "A Performance Comparison of Web Service Object Marshalling and Unmarshalling Solutions," 2011.

[19] . Z. Saurabh and D. Veronica , "JSON vs XML: A Comparative Performance Analysis Data Exchange Formats," *IJCSN International Journal of Computer Science and Network* , Vols. Volume 3, Issue 4, August 2014.

[20] E. MALIN and H. VICTOR , "Comparison between JSON and YAML for data serialization," 2011.

[21] Š. Munir, S. Muzafer and A. Emruš , "Comparative analysis of AMF, JSON and XML technologies for data transfer between the server and the client," *PERIODICALS OF ENGINEERING AND NATURAL SCIENCES,* vol. Vol. 4 No. 2 (2016).

[22] Š. Alen and M. Magdalena , "Comparison of JSON and XML Data Formats," in *Central European Conference on Information and Intelligent Systems,* September 17-19, 2014.

[23] . P. Dunlu, C. Lidong and X. Wenjie , "Using JSON for Data Exchanging in Web Service Applications," *Journal of Computational Information Systems 7: 16 (2011) 5883-5890.*

[24] . H. Zia Ul, . K. Gul Faraz and H. Tazar, "A Comprehensive analysis of XML and JSON web technologies," 2015.

[25] . P. P. Mike and . v. d. H. Willem-Jan, "Service oriented architectures: approaches, technologies and research issues," *The VLDB Journal (2007) 16:389–415.*

[26] C. RAN and S. WU , "Enterprise Information Exchange Platform Based on ESB for Research and Design," in *2010 International Conference of Information Science and Management Engineering*.

[27] W. Guanhua , "Improving Data Transmission in Web Applications via the Translation between XML and JSON," in *2011 Third International Conference on Communications and Mobile Computing*.

[28] H. Gregor and . W. Bobby, ENTERPRIS INTIGRATION PATTERNS, August 2003.

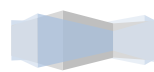
[29] *ethio telecom CBS-ETH Switch Interface Content Document*, 2018.

[30] *Ethio BSS Application Low level Design (ALLD)—CRM, Version2.1*, 1015.

[31] *Service Oriented Architecture Integration Framework Overview*, 2015.

[32] *ESB Introduction*, 2014.

[33] G. Marc and S. Neel, "Millau: an encoding format for efficient representation and exchange of XML over the Web," 2000.



Appendix

Appendix A: Java implementation result for name transformation function

run:

Name Transformation;

****

Source System Attribute:Location

TransformedData from Location to:Address

time Difference16

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix B: Java implementation result for value transformation function

run:

Value Transformation;

****

CurrentDataFormat:MM/DD/YYYY

TransformedDataFormat:17/10/2018

time Difference32

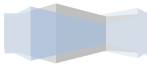
BUILD SUCCESSFUL (total time: 0 seconds)

Appendix C: Java implementation result for Attributes Aggregation

run:

AttributsAggrigation;

****



Kebele:7

Sub_city:Arada

City:Addis Ababa

Address:7Arada Addis Ababa

time Difference16

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix D: Java implementation result for Attributes splitting

run:

Attributs Splitting;

****

FullName:Yoseph Million Abye

FirstName: Yoseph

LastName: Abye

NickName: Million

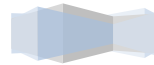
time Difference16

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix E: Java implementation result for checking complexity level, parsing XML file and the Name transformation results which is To-Be by using XML format (To-Be result)

run:
The data transformation has low complexity so used XML based data model

Start Element :root
Start Element :RequestMessage
Start Element :RequestMessageinfo
Start Element :Version
Version : 1
End Element :Version
Start Element :cbsBusinessCode



cbsBusinessCode : Payment
End Element :cbsBusinessCode
Start Element :cbsMessageSeq
cbsMessageSeq : Payment00001
End Element :cbsMessageSeq
Start Element :createdate
createdate : MM/DD/YYYY
End Element :createdate
Start Element :cbsLoginSystemCode
cbsLoginSystemCode : 101
End Element :cbsLoginSystemCode
Start Element :cbsPassword
cbsPassword : ********
End Element :cbsPassword
Start Element :arsPaymentSerialNo
arsPaymentSerialNo : 101
End Element :arsPaymentSerialNo
Start Element :arsOpType
arsOpType : 10097
End Element :arsOpType
Start Element :arcAccountKey
arcAccountKey : 201310291550
End Element :arcAccountKey
Start Element :arsPayType
arsPayType : 2
End Element :arsPayType
Start Element :arsPaymentMethod
arsPaymentMethod : 13800010036
End Element :arsPaymentMethod
Start Element :arsAmount
arsAmount : 2
End Element :arsAmount

Start Element :arsCurrencyID

arsCurrencyID : 1001

End Element :arsCurrencyID

Start Element :Banklocation

Banklocation :AdissAbaba

End Element :Banklocation

Start Element :Branch

Branch :finfna

End Element :Branch

End Element :RequestMessageinfo

End Element :RequestMessage

End Element :root

Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

time Difference109

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix F: Java implementation result for checking complexity level, converting XML format to JSON, parsing, name transformation and value transformation results (To-Be result)

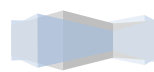
run:

The data transformation has high complexity level, so convert XML format to JSON format.

Oct 18, 2018 1:22:10 PM net.sf.json.xml.XMLSerializergetType

INFO: Using default type string

{"arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYYY","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","#text":",\n\t\t","Branch":"finfna","cbsMessageSeq":"Payment00001","arsOpType":"100



97","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

cbsLoginSystemCode : 101

Payment Result information;

****

cbsPassword : ********

arsPaymentSerialNo : 101

arsOpType : 10097

arcAccountKey : 201310291550

arsPayType : 2

arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001

Banklocation :AdissAbaba

Branch :finfna

Name Transformation;

****

Source System Attribute:Location

TransformedData from Location to:Address

Value Transformation;

****

CurrentDataFormat:MM/DD/YYYY

TransformedDataFormat:18/10/2018

time Difference125

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix G: Java implementation result for checking complexity level, converting XML format to JSON, parsing, name transformation, value transformation and attribute aggregation results (To-Be result)

run:

The data transformation has high complexity level, so convert XML format to JSON format.

Oct 18, 2018 1:12:37 PM net.sf.json.xml.XMLSerializergetType

INFO: Using default type string

{"arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYYY","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","#text":",\n\t\t","Branch":"finfna","cbsMessageSeq":"Payment00001","arsOpType":"10097","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

cbsLoginSystemCode : 101

Payment Result information;

****

cbsPassword : ********

arsPaymentSerialNo : 101

arsOpType : 10097

arcAccountKey : 201310291550

arsPayType : 2

arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001

Banklocation :AdissAbaba

Branch :finfna

Name Transformation;

****

Source System Attribute:Location

TransformedData from Location to:Address

Value Transformation;

****

CurrentDataFormat:MM/DD/YYYY

TransformedDataFormat:18/10/2018

AttributsAggrigation;

****

Kebele:7

Sub_city:Arada

City:Addis Ababa

Address:7Arada Addis Ababa

time Difference141

BUILD SUCCESSFUL (total time: 0 seconds)

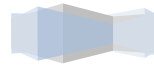
Appendix H: Java implementation result for checking complexity level, converting XML format to JSON, parsing, name transformation, value transformation, attribute aggregation and splitting results (To-Be result)

run:

The data transformation has high complexity level, so convert XML format to JSON format.

Oct 18, 2018 1:07:09 PM net.sf.json.xml.XMLSerializergetType

INFO: Using default type string



{"arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYYY","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","#text":",\n\t\t","Branch":"finfna","cbsMessageSeq":"Payment0000

1","arsOpType":"10097","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

cbsLoginSystemCode : 101

Payment Result information;

****

cbsPassword : ********

arsPaymentSerialNo : 101

arsOpType : 10097

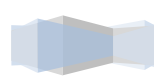
arcAccountKey : 201310291550

arsPayType : 2

arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001



Banklocation :AdissAbaba

Branch :finfna

Name Transformation;

****

Source System Attribute:Location

TransformedData from Location to:Address

Value Transformation;

****

CurrentDataFormat:MM/DD/YYYY

TransformedDataFormat:18/10/2018

AttributsAggrigation;

****

Kebele:7

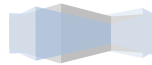
Sub_city:Arada

City:Addis Ababa

Address:7Arada Addis Ababa

Attributs Splitting;

****



FullName:Yoseph Million Abye

FirstName: Yoseph

LastName: Abye

NickName: Million

time Difference187

BUILD SUCCESSFUL (total time: 0 seconds)

AppendixI: Java implementation result for XML parsing and name transformation.

run:

Start Element :root

Start Element :RequestMessage

Start Element :RequestMessageinfo

Start Element :Version

Version : 1

End Element :Version

Start Element :cbsBusinessCode

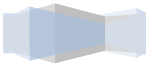
cbsBusinessCode : Payment

End Element :cbsBusinessCode

Start Element :cbsMessageSeq

cbsMessageSeq : Payment00001

End Element :cbsMessageSeq



Start Element :createdate

createdate : MM/DD/YYYY

End Element :createdate

Start Element :cbsLoginSystemCode

cbsLoginSystemCode : 101

End Element :cbsLoginSystemCode

Start Element :cbsPassword

cbsPassword : ********

End Element :cbsPassword

Start Element :arsPaymentSerialNo

arsPaymentSerialNo : 101

End Element :arsPaymentSerialNo

Start Element :arsOpType

arsOpType : 10097

End Element :arsOpType

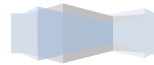
Start Element :arcAccountKey

arcAccountKey : 201310291550

End Element :arcAccountKey

Start Element :arsPayType

arsPayType : 2



End Element :arsPayType

Start Element :arsPaymentMethod

arsPaymentMethod : 13800010036

End Element :arsPaymentMethod

Start Element :arsAmount

arsAmount : 2

End Element :arsAmount

Start Element :arsCurrencyID

arsCurrencyID : 1001

End Element :arsCurrencyID

Start Element :Banklocation

Banklocation :AdissAbaba

End Element :Banklocation

Start Element :Branch

Branch :finfna

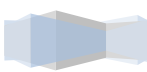
End Element :Branch

End Element :RequestMessageinfo

End Element :RequestMessage

End Element :root

Name Transformation;



****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

time Difference94

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix J: Java implementation result XML parsing, name transformation and value transformation.

run:

Start Element :root

Start Element :RequestMessage

Start Element :RequestMessageinfo

Start Element :Version

Version : 1

End Element :Version

Start Element :cbsBusinessCode

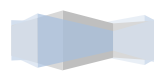
cbsBusinessCode : Payment

End Element :cbsBusinessCode

Start Element :cbsMessageSeq

cbsMessageSeq : Payment00001

End Element :cbsMessageSeq



Start Element :createdate

createdate : MM/DD/YYYY

End Element :createdate

Start Element :cbsLoginSystemCode

cbsLoginSystemCode : 101

End Element :cbsLoginSystemCode

Start Element :cbsPassword

cbsPassword : ********

End Element :cbsPassword

Start Element :arsPaymentSerialNo

arsPaymentSerialNo : 101

End Element :arsPaymentSerialNo

Start Element :arsOpType

arsOpType : 10097

End Element :arsOpType

Start Element :arcAccountKey

arcAccountKey : 201310291550

End Element :arcAccountKey

Start Element :arsPayType

arsPayType : 2

End Element :arsPayType

Start Element :arsPaymentMethod

arsPaymentMethod : 13800010036

End Element :arsPaymentMethod

Start Element :arsAmount

arsAmount : 2

End Element :arsAmount

Start Element :arsCurrencyID

arsCurrencyID : 1001

End Element :arsCurrencyID

Start Element :Banklocation

Banklocation :AdissAbaba

End Element :Banklocation

Start Element :Branch

Branch :finfna

End Element :Branch

End Element :RequestMessageinfo

End Element :RequestMessage

End Element :root

Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

Value Transformation;

****

CreatetDataFormat:MM/DD/YYYY

- TransformedDataFormat:17/10/2018

time Difference141

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix K: Java implementation result XML parsing, name transformation, value transformation and attribute aggregation

run:

Start Element :root

Start Element :RequestMessage

Start Element :RequestMessageinfo

Start Element :Version

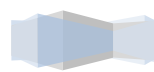
Version : 1

End Element :Version

Start Element :cbsBusinessCode

cbsBusinessCode : Payment

End Element :cbsBusinessCode



Start Element :cbsMessageSeq

cbsMessageSeq : Payment00001

End Element :cbsMessageSeq

Start Element :createdate

createdate : MM/DD/YYYY

End Element :createdate

Start Element :cbsLoginSystemCode

cbsLoginSystemCode : 101

End Element :cbsLoginSystemCode

Start Element :cbsPassword

cbsPassword : ********

End Element :cbsPassword

Start Element :arsPaymentSerialNo

arsPaymentSerialNo : 101

End Element :arsPaymentSerialNo

Start Element :arsOpType

arsOpType : 10097

End Element :arsOpType

Start Element :arcAccountKey

arcAccountKey : 201310291550

End Element :arcAccountKey

Start Element :arsPayType

arsPayType : 2

End Element :arsPayType

Start Element :arsPaymentMethod

arsPaymentMethod : 13800010036

End Element :arsPaymentMethod

Start Element :arsAmount

arsAmount : 2

End Element :arsAmount

Start Element :arsCurrencyID

arsCurrencyID : 1001

End Element :arsCurrencyID

Start Element :Banklocation

Banklocation :AdissAbaba

End Element :Banklocation

Start Element :Branch

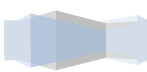
Branch :finfna

End Element :Branch

End Element :RequestMessageinfo

End Element :RequestMessage

End Element :root



Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

Value Transformation;

****

CreatetDataFormat:MM/DD/YYYY

TransformedDataFormat:17/10/2018

AttributsAggrigation;

****

Banklocation:AdissAbaba

Branch:finfna

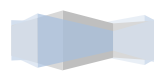
Address:AdissAbabafinfna

time Difference172

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix L: Java implementation result XML parsing, name transformation, value transformation and attribute aggregation and splitting results.

run:
Start Element :root
Start Element :RequestMessage
Start Element :RequestMessageinfo
Start Element :Version



Version : 1

End Element :Version

Start Element :cbsBusinessCode

cbsBusinessCode : Payment

End Element :cbsBusinessCode

Start Element :cbsMessageSeq

cbsMessageSeq : Payment00001

End Element :cbsMessageSeq

Start Element :createdate

createdate : MM/DD/YYYY

End Element :createdate

Start Element :cbsLoginSystemCode

cbsLoginSystemCode : 101

End Element :cbsLoginSystemCode

Start Element :cbsPassword

cbsPassword : ********

End Element :cbsPassword

Start Element :arsPaymentSerialNo

arsPaymentSerialNo : 101

End Element :arsPaymentSerialNo

Start Element :arsOpType

arsOpType : 10097

End Element :arsOpType

Start Element :arcAccountKey

arcAccountKey : 201310291550

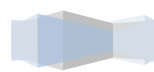
End Element :arcAccountKey

Start Element :arsPayType

arsPayType : 2

End Element :arsPayType

Start Element :arsPaymentMethod



arsPaymentMethod : 13800010036

End Element :arsPaymentMethod

Start Element :arsAmount

arsAmount : 2

End Element :arsAmount

Start Element :arsCurrencyID

arsCurrencyID : 1001

End Element :arsCurrencyID

Start Element :Banklocation

Banklocation :AdissAbaba

End Element :Banklocation

Start Element :Branch

Branch :finfna

End Element :Branch

End Element :RequestMessageinfo

End Element :RequestMessage

End Element :root

Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

Value Transformation;

****

CreatetDataFormat:MM/DD/YYYY

TransformedDataFormat:17/10/2018

AttributsAggrigation;

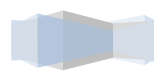
****

Banklocation:AdissAbaba

Branch:finfna

Address:AdissAbabafinfna

Attributs Splitting;



****

BalanceInfo:C_MAIN_BILLING_ACCOUNT 10002345

BalanceName: C_MAIN_BILLING_ACCOUNT

BalanceId: 10002345

time Difference203

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix M: Java implementation result JSON parsing and name transformation (As-Is JSON)

run:

{"BalanceInfo":"C_MAIN_BILLING_ACCOUNT10002345","arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYYY","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","Branch":"finfna","cbsMessageSeq":"Payment00001","arsOpType":"10097","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

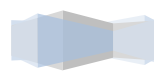
cbsLoginSystemCode : 101

BalanceInfo : C_MAIN_BILLING_ACCOUNT10002345

Payment Result information;

****

cbsPassword : ********



arsPaymentSerialNo : 101

arsOpType : 10097

arcAccountKey : 201310291550

arsPayType : 2

arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001

Banklocation :AdissAbaba

Branch :finfna

Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

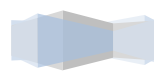
time Difference78

BUILD SUCCESSFUL (total time: 2 seconds)

Appendix N: Java implementation result JSON parsing, name transformation and value transformation (As-Is JSON)

run:

{"BalanceInfo":"C_MAIN_BILLING_ACCOUNT10002345","arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYY



Y","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","Branch":"finfna ","cbsMessageSeq":"Payment00001","arsOpType":"10097","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

cbsLoginSystemCode : 101

BalanceInfo : C_MAIN_BILLING_ACCOUNT10002345

Payment Result information;

****

cbsPassword : ********

arsPaymentSerialNo : 101

arsOpType : 10097

arcAccountKey : 201310291550

arsPayType : 2

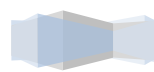
arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001

Banklocation :AdissAbaba

Branch :finfna



Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

Value Transformation;

****

CreatetDataFormat:MM/DD/YYYY

TransformedDataFormat:17/10/2018

time Difference93

Appendix O: Java implementation result JSON parsing, name transformation, value transformation and attribute aggregation (As-Is JSON)

run:

{"BalanceInfo":"C_MAIN_BILLING_ACCOUNT10002345","arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYYY","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","Branch":"finfna ","cbsMessageSeq":"Payment00001","arsOpType":"10097","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

cbsLoginSystemCode : 101

BalanceInfo : C_MAIN_BILLING_ACCOUNT10002345

Payment Result information;

****

cbsPassword : ********

arsPaymentSerialNo : 101

arsOpType : 10097

arcAccountKey : 201310291550

arsPayType : 2

arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001

Banklocation :AdissAbaba

Branch :finfna

Name Transformation;

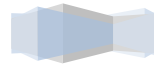
****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

Value Transformation;

****



CreatetDataFormat:MM/DD/YYYY

TransformedDataFormat:17/10/2018

AttributsAggrigation;

****

Banklocation:AdissAbaba

Branch:finfna

Address:AdissAbabafinfna

time Difference110

BUILD SUCCESSFUL (total time: 0 seconds)

Appendix P: Java implementation result JSON parsing, name transformation, value transformation and attribute aggregation and splitting results (As-Is JSON)

run:

{"BalanceInfo":"C_MAIN_BILLING_ACCOUNT10002345","arcAccountKey":"201310291550","arsAmount":"2","cbsBusinessCode":"Payment","arsPaymentSerialNo":"101","createdate":"MM\/DD\/YYYY","cbsVersion":"1","cbsLoginSystemCode":"101","arsCurrencyID":"1001","Branch":"finfna","cbsMessageSeq":"Payment00001","arsOpType":"10097","arsPaymentMethod":"13800010036","Banklocation":"AdissAbaba","arsPayType":"2","cbsPassword":"********"}

Request Message;

****

cbsVersion : 1

cbsBusinessCode : Payment

cbsMessageSeq : Payment00001

cbsLoginSystemCode : 101

BalanceInfo : C_MAIN_BILLING_ACCOUNT10002345

Payment Result information;

****

cbsPassword : ********

arsPaymentSerialNo : 101

arsOpType : 10097

arcAccountKey : 201310291550

arsPayType : 2

arsPaymentMethod : 13800010036

arsAmount : 2

arsCurrencyID : 1001

Banklocation :AdissAbaba

Branch :finfna

Name Transformation;

****

Source System Attribute:cbsMessageSeq

TransformedData from cbsMessageSeqto:cbsMsgLanguageCode

Value Transformation;

****

CreatetDataFormat:MM/DD/YYYY

TransformedDataFormat:17/10/2018

AttributsAggrigation;

****

Banklocation:AdissAbaba

Branch:finfna

Address:AdissAbabafinfna

Attributs Splitting;

****

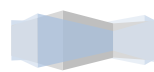
BalanceInfo:C_MAIN_BILLING_ACCOUNT 10002345

BalanceName: C_MAIN_BILLING_ACCOUNT

BalanceId: 10002345

time Difference141

BUILD SUCCESSFUL (total time: 0 seconds)



Declaration

I, the undersigned, declare that this MSc thesis is my original work, has not been presented for fulfillment of a degree in this or any other university, and all sources and materials used for the thesis have been acknowledged.

Alemtsehay Kebede
Name

_____
Signature

Place: Addis Ababa, Ethiopia

Date of submission: October 19, 2018

This thesis has been submitted for examination with my approval as a university advisor.

Dr.Mesfin Kifle
Advisor's Name

_____
Signature