Sudan University of Science and Technology

College of Post Graduate Studies

A Thesis Submitted in Partial Fulfillment of the Requirements of M.Sc. in Computer Science

Evaluating the Quality of Integrated Software Systems Based on Service-Oriented Architecture

تقويم جودة أنظمة البرمجيات المتكاملة المعتمدة علي معمارية البنيه الخدمية

Prepared By
Ayman Amin Ali Amin

The Supervision
Dr. Nisreen Beshir Osman

April 2018

الآية

﴿ ألم تَرَ أنَّ الله أنزلَ من السَّماء ماءً فأخرجنا به ثمراتٍ مُخْتلفاً ألوانُها ومن الجبالِ جُدَدٌ بيضٌ وحمرٌ مختلفٌ ألوانُها وغرابيبُ سُودٌ ومن النَّاسِ والدَّوابِّ والأنعامِ مُختلفٌ ألوانُهُ كذلك إنَّما يَخشى الله من عبادهِ العلماءُ إنَّ الله عزيزٌ غفور﴾

(35 فاطر آية 27-28).

Dedication

I dedicate this thesis to my parents, Amin Ali Amin and Omaima Abdalhay who supported me to continue my educational process, to my colleagues and finally to my teacher Ashwaq who gave me advices that helped me in my thesis.

Abstract

Most current systems operate independently of each other. When the organization needs a comprehensive system, it dispenses the existing systems to build a new one. Such strategy wastes resources and time. The integration of existing systems saves time, resources and it prevents risks in the new system. Moreover, using the quality model ensures that the system after integration is working correctly. The main objective of this research is to propose a model that could be used to evaluate the quality of integrated systems. The proposed model specifies the attributes that determine the quality of integrated systems. In order to measure these attributes a set of metrics were identified such as Existence of meta-information, Mean Time to Repair, and Mean Time between Failures**.** In order to validate the model two systems were selected and integrated using SOA technology. The results of the evaluation process showed high rate for the reusability, correctness, and reliability, normal rate for the usability, performance and modifiability.

المستخلص

تعمل معظم النظم الحالية بشكل مستقل عن بعضها البعض، وعندما تحتاج المؤسسة إلى نظام شامل، فإنها تقوم ببناء نظام جديد ،وهذه الإستراتيجية تهدر الموارد والوقت،كما يمكن ربط النظم التي تعمل بشكل منفصل وذلك يوفر الوقت والموارد ويمنع المخاطر الناتجه في عملية بناء النظام الجديد ، وبالإضافة الي ذلك يمكن إستخدام نموذج قياس للجودة لنضمن أن النظام الناتج من عملية الربط يعمل بشكل صحيح ، الهدف الرئيسي من هذا البحث هو إقتراح نموذج يمكن إستخدامه لتقييم جودة النظم المتكاملة، في إطار ذلك يحدد النموذج المقترح السمات التي تقيس جودة الأنظمة المربوطة .من أجل قياس هذه السمات تم تحديد مجموعة من المقاييس مثل وجود المعلومات الوصفية، ومتوسط الوقت للإصلاح، ومتوسط الوقت بين الفشل وغيرها من المقاييس ، من أجل التحقق من صحة النموذج تم إختيار نظامين ومن ثم ربطهما بإستخدام تقنية البنية الخدمية (SOA) وأظهرت نتائج عملية التقييم معدل عالي في كل من إعادة الاستخدام، والصحة، والموثوقية، ومعدل الطبيعي (المتوسط) لسهولة الإستخدام والأداء وقابلية التعديل.

Table of contents

Table of figure

Table of tables

Table of terms

Abbreviation	Terms
SOA	Services oriented architecture
IEEE	Institute of Electrical and Electronics Engineers
IDLs	Interface Definition Languages
EAI	Enterprise Application Integration
HTTP	Hypertext transfer protocol
XML	Extensible Markup Language
WSDL	Web Services Description Language
SaaS	Software as a Service
CC	Cloud Computing
COTS	Commercial off-the-shelf
APIs	Application-programming interfaces
SynCSI	Syntactic Completeness of Service Interface
SemCSI	Semantic Completeness of Service Interface
EMI	Existence of meta-information
SCSr	Self-Completeness of Service's Return Value
SCSp	Self-Completeness of Service's Parameter
MTTR	Mean Time To Repair
MTBF	Mean Time Between Failures
SRT	Service Response Time
TP SRV	Throughput of Service
TC per M	Number of test case per a method or function
KLOC	Thousands of lines of code

Chapter One

Introduction

1.1 Introduction

The rapid development of science and technology, helped to produce many systems inside or outside enterprises. However, many of these systems still work separately from each other.

The service oriented architecture (SOA) is not just a technology of integration, but it is also an architecture to guide the process of analysis and development life cycle. SOA is an architecture where different services in different systems are connected together to build a new system [1]. In reality, there are many software quality models that used to assess the quality of target system [2] based on specific attribute. We built model dedicated to evaluate the integration of systems.

1.2 Problem Statement

Recently, there are many ready sub-systems in organizations and market. these systems may perform just some part of organization's functions separately. Because of this separation in performance, there is a tedious manual work to pass the data from one sub-system as output to another one as input (waste of time and effort).

There are many models to evaluate the quality systems, but there is no such a model that is dedicated to evaluate the quality of integrated systems after integration process.

1.3 Research Significance

Most current systems operate independently of each other. when an organization needs a comprehensive system it dispenses the existing systems to build a new one. Such strategy wastes resources and time. Integrating the existing systems saves time, resources and it prevents new risks in the new system. Moreover, using the quality model ensures that the system after integration works correctly and the staff is satisfied.

1.4 Research Objective

The main objective of this research is evaluating the integrated systems by building a quality model for system integration.

In order to build the model, the following sub-objective needs to be achieved:

1. Select the quality attributes that used in evaluation .
2. Identify the metrics of each attribute.

1.5 Research Scope

This is study focus on building a quality model dedicated to evaluate the integration of sub-systems based on service-oriented architecture (SOA) technique. Taking into consideration the attributes that related to the concept of SOA and integration.

1.6 Research Organization

The structure of this research divided into four chapters as shown below:

- **Chapter one**: Introduction

  This chapter describes the whole idea behind the theses. Defines the problem statement, why it is important, objective and scope research.

- **Chapter two**: Literature review

  Divided into two section. First one takes the major concept about topic and the second one discusses the previous studies.

- **Chapter three**: Methodology

  This chapter represents the methodology of research. The contents are: the way of building quality model, extract metrics and integrate the selected sub-systems.

- **Chapter four**: Results

  Contents the results of applying the quality model to the final system.
- **Chapter five**: Conclusions and Recommendations
- **References.**

Chapter Two

Literature Review & Related Work

2.1 Introduction

This chapter is divided into two sections. The first section gives general description about integration, quality model and service oriented architecture. The second section describes the related studies to research project.

2.2 Literature Review

2.2.1 Integration

2.2.1.1 Definition

Melding existing system and new technologies to form capable systems that intended to take additional tasks, exhibit improved performance and/or enhance existing systems [3].

The IEEE Standard Glossary of Software Engineering Terminology [5] defines integration as “the process of combining software components, hardware components, or both, into an overall system”.

2.2.1.2 Objectives for Systems Integration Methodology

The objectives for a systems integration engineering methodology can be stated as following [3]:

- To provide a suitable methodology that encompasses the entire integration program. Starting from requirements, moving to design, construction, test, and finally to deployment and maintenance.
- Facilitate understanding and communication.
- To enable capture of design and implementation needs early, especially interface and interactive needs associated with bringing together new and existing equipment and software.
- To support both top-down and bottom-up design philosophy.

- To support full compliance with audit trail needs, system-level quality assurance, and risk assessment and evaluation.

2.2.1.3 Integration challenge

Challenges for achieving integration mostly have to do with the inherent difficulties of linking a series of diverse existing systems that may be produced by multiple different manufacturers. Other integration challenges have to do with the lack of a coherent or unifying data structure that links all of the different systems, an unwieldy framework of multiple different applications and support systems and the sheer age of the different systems and the actual delivery of the information to key business units that need it. These integration challenges hinder overall process efficiency because poor data exchange between systems prevents quick communication among business units.

2.2.1.4 Typical Systems Integration Life-cycle Phases

The most commonly seven-phase life cycle used in SI programs is as following [3]:

- Requirements definition and specification.
- Feasibility analysis.
- System architecture development.
- Management plan: program and project plan.
- Systems design logical and physical design.
- Implementation: design implementation, system tests, and operational deployment.
- Evaluation: system review and plan for replacement/retirement.

2.2.1.5 Existing Approaches to Software Integration

1. Standard Interfaces and Open Systems

The common understanding of an “open” system is that it should e.g. be portable, scalable, and interoperable through means of standard interface. The

importance of standards applies not only to interfaces but also to domain-specific architectures as well [4].

2. **Component-Based Software**

The integration context of component-based software is when there are pre-existing software components with clearly defined interfaces available for integration. A component-based approach can be used even with systems that completely developed in-house [4].

Interface Definition Languages (IDLs) are central part of component technology, and integration at the function call level is relatively straightforward [4].

3. **Enterprise Application Integration (EAI)**

EAI defined as the process of integrating enterprise systems with existing applications [6].

2.2.2 Service Oriented Architecture (SOA)

2.2.2.1 Definition:

SOA is a business-centric IT architectural approach that supports integrating business as linked, repeatable business tasks, or services [7].

With SOA, the business logic is decomposed into well-defined and reusable services, which will have exposed for everyone to use. Now all the application has to do is to consume them. As such. Now the application code is reduced greatly.

2.2.2.2 Web Services

Web service is a realization of SOA and it is the most popular SOA implementation [7].

Web service terminologies [7]:

- Hypertext transfer protocol [HTTP]

HTTP is a widely accepted standard that is implemented in many systems. By using the HTTPS protocol, web service communication will be secured.

- Extensible Markup Language [XML]

XML chosen, as it's a platform-independent language that can be understand by different systems.

- Web Services Description Language [WSDL]

WSDL typically includes where the service is located, the functions/methods available for invocation, parameters and its data type as well as the format of the response.

2.2.3 Software Quality Models

2.2.3.1 What is the quality?

Defined from two perspectives first one, conformance to specification: Quality that defined as a matter of products and services whose measurable characteristics satisfy a fixed specification – that is, conformance to an in beforehand defined specification [10]. **The second view, meeting customer needs**: Quality that is identified independent of any measurable characteristics. That is, quality defined as the products or services capability to meet customer expectations – explicit or not [10].

2.2.3.2 Definition of Quality Model

A quality model is a set of characteristics and sub-characteristics, as well as the relationships between them that provide the basis for specifying quality requirements and for evaluating quality of the component or the system [10].

Measurement of quality attributes is concerned with deriving the numerical values by using the appropriate metrics for that attribute [10].

2.2.3.3 Software Quality Assurance – Defined Below

IEEE [9] **defined Software quality assurance as** a planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements. A set of activities designed to evaluate the process by which the products developed or manufactured. Contrast with quality control. **Another general definition suggested for the software quality assurance [9] as** a systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines [9].

2.2.4 Quality Factor

The typical objective of a quality factor is to characterize an aspect of the quality of a work product or a process [11].

2.2.4.1 Quality Sub-factor

Some factors cannot refer directly to their criteria; they require an extra intermediate level to compute. Elements of this intermediate level are sub-factors. For example, in Boehm's model, maintainability as factor refers to three sub-factors: Testability, understandability, and modifiability [11].

The typical objectives of a quality sub-factor are to [11]

- Characterize a part of a quality factor.
- Further, characterize an aspect of the quality of a work product or process.
- Help in defining the term “quality” for an endeavor.

2.2.5 Quality Metrics

2.2.5.1 Definition

A metric is a quantifiable measurement of software product, process, or project that directly observed, calculated, or predicted [12].

2.2.5.2 Main objectives of software quality metrics

- To facilitate management control as well as planning and execution of the appropriate managerial interventions. Achievement of this objective based on calculation of metrics regarding [11]:
    - Deviations of actual functional (quality) performance from planned performance.
    - Deviations of actual timetable and budget performance from planned performance.
- To identify situations that require or enable development or maintenance process improvement in the form of preventive or corrective actions introduced throughout the organization. Achievement of this objective based on Accumulation of metrics information regarding the performance of teams, units, etc. [11]

An approach to quality is to decompose quality in Factors, Sub-factors, and criteria. Evaluation of a program begins with measuring each quality criteria with numerical value from metrics. Then, each quality sub-factors assessed using their criteria. Finally, numerical value assigned to quality characteristics from their quality sub factors [11]. See figure (2.1).

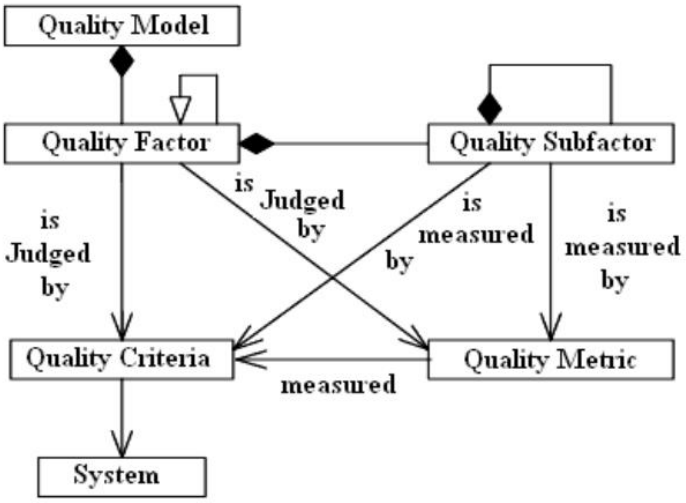


Figure (2.1) Relationship among quality model elements [10].

2.3 Related Work

2.3.1 A Quality Model for Evaluating Reusability of Services in SOA

Si Won Choi et al. [14] evaluated the SOA based on reusability, the "reusability of services is a key criterion for evaluating the quality of services". They extracted the key feature of service in SOA (Commonality of Services, Modularity of Services, well-defined interface, Loosely-Couple Nature, Heterogeneity, Standardization, Subscription-based Invocation, evolve ability of Service, Limited Manageability, Partial Matching and Adaptability, Business Aligned) [14]. They defined the reusability of service as "the degree to which the service can be used in more than one business process or service application, without having much overhead to discover, configure, and invoke it". They derived quality attributes from the key features of services, as shown in figure (2.2)

The solid arrow indicates Strongly Derives mapping and the dashed arrow indicates Weakly Derives mapping see Figure (2.2) [14].

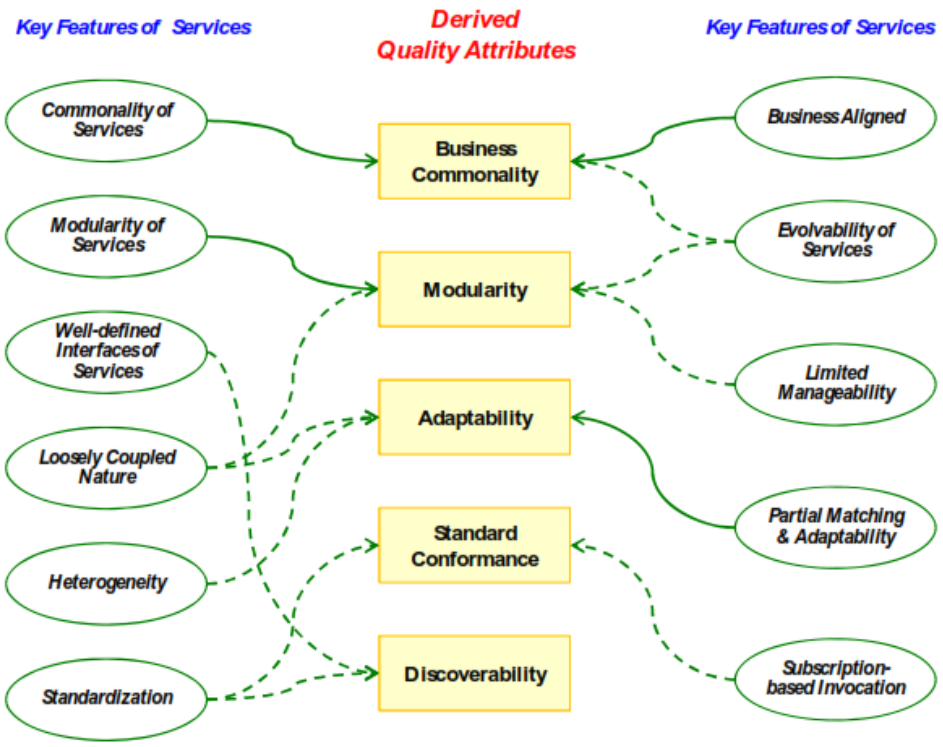


Figure (2.2) the relation between feature and quality attributes [14].

- They defined metrics for each quality attribute.

2.3.2 A Quality Model for Evaluating Software-as-a-Service in Cloud Computing

Lee et al. [15] proposed a comprehensive model for evaluating quality of SaaS. In the first they defined key features of SaaS (Reusability, Data Managed by Provider, Service Customizability, Availability, Scalability, pay per Use) based on CC (Cloud Computing). Then, they derived quality attributes from the key features. Based on ISO/IEC 9126, Efficiency and Reliability extended to cover CC specific features. Derived from the key features, Reusability, Availability, and Scalability are newly defined see Figure (2.3) [15].

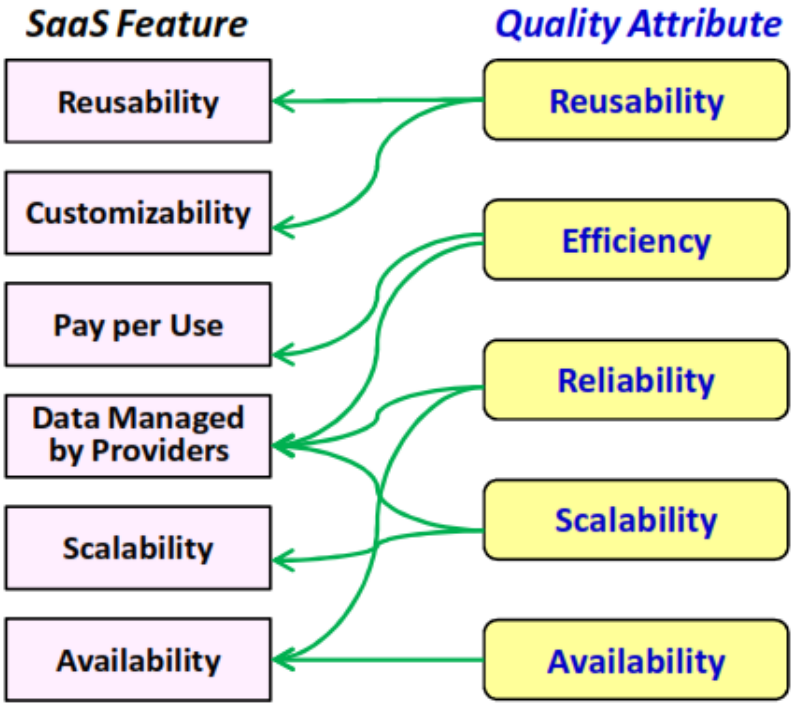


Figure (2.3) Mapping from Feature to Quality Attributes [15].

Finally, defined metrics for this quality's attributes [15].

2.3.3 A Software Quality Model for SOA

Goeb et al. [16] introduced a unifying notion to model (a meta-model) the quality of SOA systems. Its leads to a trisection of concepts into goals, facts and measures: To evaluate the achievement level for defined quality goals, certain system properties (facts) considered which in turn quantified using measures. They Introduces quality-related activities used in the model (Understanding, Consumption, Support, Service Reuse, and Extension) with explain of any one. They presented some factors that influence quality relevant activities (some of them not all, Because of the limited space) [16]. Finally, they showed number of measures in the SOA quality model and each measure identified by a unique ID. For the same reason they not presented all measures for all factor in the model [16].

2.3.4 Quality Assurance and Integration Testing Aspects in Web Based Applications

Khan et al. [17] wanted to make integration testing for web application and suggested the following major characteristics of different web applications like [17]:

- Network intensive: Web applications delivered to a diverse community of users.
- Content: It is heavily content-driven because it has different content textual, graphic and so on.
- Continuous evolution: These applications updated on the regular interval, even some applications updated on hourly.
- Short development schedule: These applications have very short time for the development.
- Security: Because there is different people are use web application. It is difficult to keep data secure with regular method of security.
- Aesthetic.

Some other characteristics such as distributed, heterogeneity, autonomous, dynamic, hypermedia, and multiplatform support, ubiquitous are very important to understand [17]. They said the quality is satisfaction of customer base on According to IEEE. **"Customer Satisfaction = Compliant Product + Good Quality + Delivery within Schedule and Budget" [17].** They used integration testing because it is sure that modules and their interfaces in an application interact with each other in a correct and secure way [17]. Integration testing covers following types of concerning areas during integrating different modules [17]:

- Calls of different software components, while interacting to each other.
- Data and information sharing between the modules in proper manners.
- Compatibility, which ensures one module that does not effect on the performance and functionality of the other modules.

- Non-functional issues.

The integration testing it became difficult for different reason like (large size, multilingual components and use of different operating systems). However, the most important and commonly known challenges during integration testing are [17]:

- Heterogeneous Infrastructure and Environment: The assurance of compatibility and interoperability between these components is one of major concern during the testing process.
- Service Oriented Environment: Web services and SOA demand very explicit inputs and outputs. In this environment many applications send only update or received messages, there is no guarantee that data came from system A to system B is accurate.
- Heterogeneous Database.
- Inconsistent Interaction Models: Control protocols are responsible of defining rules that how integrated components interact to each other. Data models define the contents and format of communication between them.
- Distributed Nature of Systems: Distributed nature of systems can have great impact on working of Web-based applications and these issues can solved during integration testing.

2.3.5 A survey on Software as a service (SaaS) Using Quality Model in Cloud Computing

Dehmi Kalan et al. [18] tried to build comprehensive quality model for evaluation SaaS. In first step they extracted key features of SaaS from them evaluation on current references in cloud computing. See the figure (2.4) [18].

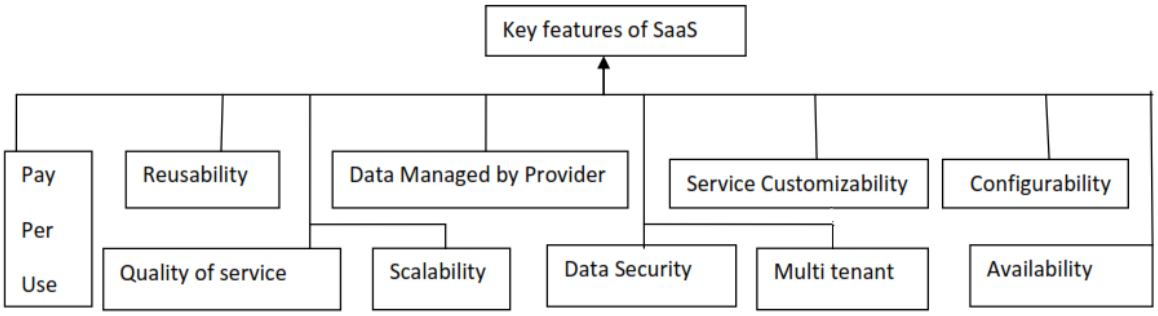


Figure (2.4) the key feature of SaaS [18].

The quality model that they proposed consist of three factors, i.e. security, quality of service and software quality. As SaaS, service involved three roles or perspective (customer, platform and application developer). Each quality factor categorized into three parts. They took Security with three roles like (Customer Security, Application Security and Network Security) [18].

2.3.6 A New Software Quality Model for Evaluating COTS (Commercial of the Shelf) Components

Rawashdeh et al. [19] studied different type of quality model. Hierarchy models like (McCall's, Boehm's, FURPS, and ISO/IEC 9126) and non-hierarchy models like (Triangle and Quality Cube). After that, they mixed between them to build comprehensive model that covering the different perspective for different stakeholder [19]. For methodology, at first they defined small set of attributes (Functionality, Reliability, Usability, Efficiency, Maintainability, and Manageability). Then Distinguish between internal and external metrics. For COTS components. As next step they Identify Stakeholders (type of users) for each high-level quality attribute. Lastly, they put the pieces together as new quality model [19].

- The distinction between internal and external metrics led us to realize that external metrics is more appropriate for COTS components [19].

- The base of new model was ISO 9126 because it includes the common software quality characteristics that supported by the other model [19].

2.3.7 Integration architecture: Comparing web APIs with service-oriented architecture and enterprise application integration

Clark et al. [20] say the web APIs technique considered to be better than other two techniques (Application-programming interfaces (APIs), enterprise application integration), but it is important to recognize that the patterns, techniques, and concepts developed along the way, such as hub and spoke architecture and SOA, are still relevant and appropriate in the right circumstances [13]. The SOA is better than Application programming interfaces (APIs) because it more secure and we will hide the details of code and procedure from consumer and we just pass parameter via URL. In addition, the researchers mentioned that the SOA increase business ownership, decoupling and broadening audience more than other techniques [13].

2.3.8 Discus of Previous Studies

Based on the previous studies and our research on internet, we found there is no quality model dedicated to evaluate the point of integrated system but there are many models to evaluate the technique of integration like (SOA). Those models that are evaluate the techniques not cover all attributes that are related to integration concept. This is research used the existing attributes in research that is related to the SOA and expanded by adding new attributes to cover all aspect of integration.

Chapter Three

Methodology

3.1 Introduction

At this chapter, we built the Quality Model for integrated Systems by selecting the quality attributes or sub quality attributes. These attributes are related to integration and SOA concept. In addition, we extracted the metrics of these attributes.

3.2 Methodology

There’re no quality models dedicated to the integrated systems, but there’re some studies talking about some quality attributes related to SOA. and the effect of the SOA when we use it. Therefore, the research idea concerned about building quality model to evaluate the integrated systems.

In order to build the model used to evaluate quality of integrated system the following steps need to be followed see figure (3.1)

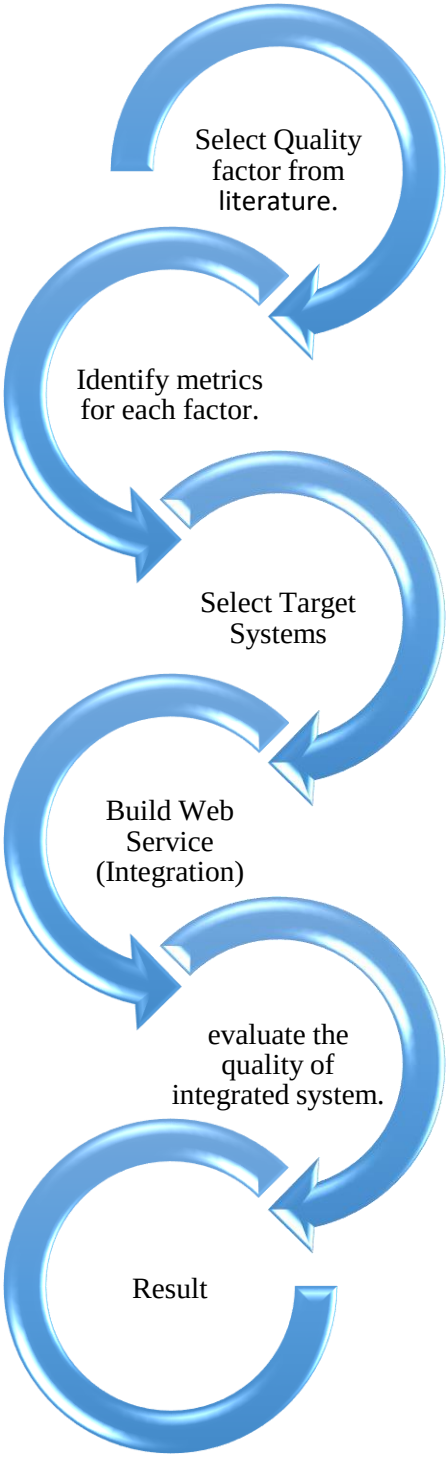


Figure (3.1) the research methodology.

3.2.1 Select Quality Factor

To build a model which contains a number of quality factors that selected from literature review to suit the systems that will be integrated. The quality factors that have been selected based on SOA, as well as quality factors that affected by the integration, which are mentioned below

❖ **Usability** Interfaces ware well-built and described [19]. **Reusability of** services can be considered as a key criterion for evaluating the quality of services [15].

❖ **Reliability** A system that built using service-oriented architecture and provides many services will work well even if there's failure occurs in a specific service. the rest of services never stop [15].

❖ **Performance** Means different things in different contexts [17] **Response time,** how long does it take to process a request? **Throughput** how many requests overall can be processed per unit of time. **Timeliness** ability to meet deadlines.

❖ **Maintainability** covers two main attributes [19] (Modifiability and Testability) **Extensibility** is a special case of **Modifiability**. Capabilities can be extended without affecting other parts of the system (services). **Testability** Testing is the root of quality. all services can be tested to detect any kind of errors.

❖ **Correctness** Means these services or interfaces we built are doing the right things that we built them to [21].

3.2.2 Identify Metrics for Each Factor

3.2.2.1 Usability and Reusability

In this case, we found out that the usability and reusability are concerned with information that related with web services that mentioned in WSDL. We consider that usability includes reusability since that if the web services are usable then they are already reusable.

1. Usability

Is the measure of ease and effectiveness when users use the service or/and assemble it to their systems [2].

❖ There are two metrics for the usability. The **Syntactic** and the **Semantic** Completeness of Service Interface. The completeness means how many well described service operations are there in the service interface [20].

- **Syntactic Completeness of Service Interface (SynCSI)**

The syntactic elements indicate to the contents of service operations [20].

$$\text{SynCSI} = \frac{\text{Well Described Syntactic Elements}}{\text{Total Number of Syntactic Elements In WSDL}}$$

- **Semantic Completeness of Service Interface (SemCSI)**

The semantic elements indicate to the contents of semantic information of service operation. such as pre condition, post condition, constraint and so on. There is no standard way to describe the semantic of services. It is still in research area [20].

$$\text{SynCSI} = \frac{\text{Well Described Semantic Elements}}{\text{Total Number of Semantic Elements In WSDL}}$$

2. Reusability

SOA developed with a hope to enhance the reusability of services. Due the ability to work with independence for specific environment. The selected metrics of the reusability based on reuse metrics for component. According to the five metrics that used to measure the component based. Three of them are relevant to web service [19].

These are the selected metrics

- **Existence of meta-information (EMI)**

This is a specification that describes functions, protocol and interface of service. Value of metric is zero when there is no specification and one if there is a specification provided [19].

- **Self-Completeness of Service's Return Value (SCSr)**

If a business method has no return value, it will not relate with objects that use it. Therefore, this method is more independent and has higher portability [19].

$$SCSr(S) = \frac{\sum Mbr}{Mb}$$

Where

- **Mbr** is sum of methods that have not return value.
- **Mb** is the sum of method in an encapsulated unit.

- **Self-Completeness of Service's Parameter (SCSp)**

If a business method has no argument, it will not depend on objects that use it [19].

$$SCSp(S) = \frac{\sum Mbp}{Mb}$$

Where

- **Mbp** is sum of methods that have not argument.
- **Mb** is the sum of method in an encapsulated unit.

3.2.2.2 Reliability

To measure the reliability, two attributes are used: recoverability and availability. They focus on repair the system as soon as possible after failure.

We measured two metrics

- **Mean Time to Repair (MTTR)**

The unit of measurement for this metrics will be seconds, minutes or hours. It is the time between two failures.

Mean Time to Repair = (Total down time) / (number of breakdowns).

- **Mean Time Between Failures (MTBF)**

The value of MTBF will be always more than zero. MTBF = summations of time between failures / total number of failures. Alternatively, MTBF = (Total up time) / (number of breakdowns).

3.2.2.3 Performance

Measuring the performance introduces three characteristic. Response time, Throughput and Timeliness.

- **Service Response Time (SRT)**

The metrics of SRT is the time between service request ending and request response beginning [20].

The equation

SRT = Time when Service Consumer finishes sending request to the service – Time when Service Consumer starts receiving response from the service

- The lower of SRT indicator to higher response time of service.

- **Throughput of Service (TP SRV)**

Number of requests served at specific period [20].

The equation

TP (SRV) = Number of Complete Service Requests / Unit of Time.

- The Higher of (TP SRV) indicator to better performance.

- Unit of Time can be measure by second, minute, hour…

- **Timeliness**

The measure of this attribute can be **Yes** or **No.** if they meet the deadlines of process (request) or not [20].

The real time request <= the deadline time we put

3.2.2.4 Maintainability

This is a characteristic for feature revision. We get the benefits of it after finish. It's divided into two attributes modifiability and testability.

1. Modifiability

The modifiability attribute has many sub attributes that can be measured. However, this model only covers the extensibility as sub attribute because it is the only attribute that effected by SOA and the concept of integration. There is a need to extend the services to get more benefits of collaboration of sub system [22].

1.1 Extensibility

This attribute represents the number of interfaces (services) that can be built and added to our system [22].

Metrics

Number of interfaces that we add to our systems to integrate (# interface).

2. Testability

The target of this attribute is to test validation of built interface. Also it can be used to test the future modification. The target of this test to get zero defect [4].

In case of web service, this attribute tries to check all functions using different cases of input (date type, wrong value and true value even that may cause exception). The metrics showed below is going to measure the testability.

Metrics

Number of test case per a method or function (**TC per M**). The unit of measurement for this metrics will be test case/method.

3.2.2.5 Correctness

Correctness can be defined as the degree of which the system (services) performs the specified function. Alternatively, the degree of system (services) to satisfy the specifications or user's mission objectives [21].

Metrics

- Defects per KLOC - most common measure for correctness (Defects per KLOC).
- Defects ware counted over a specific period (Defects per period).
- Number of non-conformance service / total number of services (percentage of Non-CS) [21].

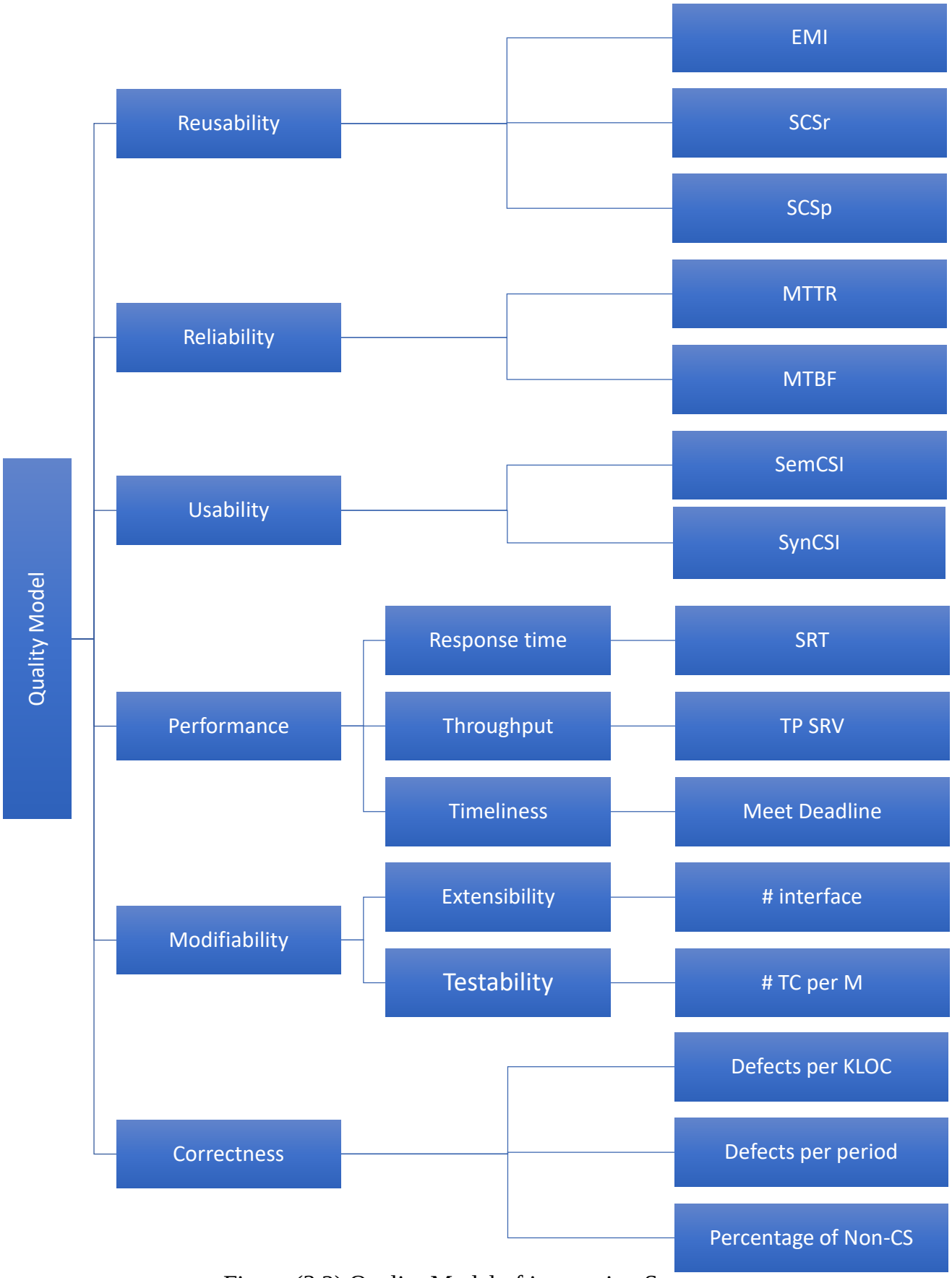


Figure (3.2) Quality Model of integration Systems

3.2.3 Select Target Systems

Midlife company is a Medicine Company. They have comprehensive system used to manage the internal process that occur within the company. They support many pharmacies by medicine. Those pharmacies either have their own systems or they want to buy new one. In this context, there will be external process that occur between midlife and other pharmacies such as exchange information about medicine (Name, Barcode and so on) or information related to insurance company that stored in midlife such as percentage of Lowering on medicine, people or for both. Probably the two sides will be both consumer and provider services.

We selected the midlife's system as first system. On other side, we selected one of pharmacy's systems. We want to integrate these two systems to build one system that can increase the connectivity between the two sides and get the goals of midlife.

3.2.4 Build Web Service (Integration)

The web service built based on java programing language, the midlife's system developed using PHP language and the pharmacy system developed by C# one of .NET programing language. NetBeans IDE is a tool that we used to build the web services.

We built web services to integrate the selected system at specific points. We built four functions one of them especially for medicine to get all information about medicine from midlife's database to be unified for all pharmacies, and the other three functions related to insurance information.

These are the ordered steps to be followed in building the web service:

1- Select the programming language that will be used to build web service.
2- Select the IDE based on language.
3- Understand the two systems and identify the points of integration.
4- Build the web service as server side using soap techniques.

5- Use the web service interface that built in two systems to transfer the data.

3.2.5 Apply the Model on Systems

The final version of the system after integration considered as case study to experiment our quality model. We used the metrics questions or equation to get the results. Chapter four is going to discuss the result of evaluation in more details.

Chapter Four

Model implementation & Result

4.1 Introduction

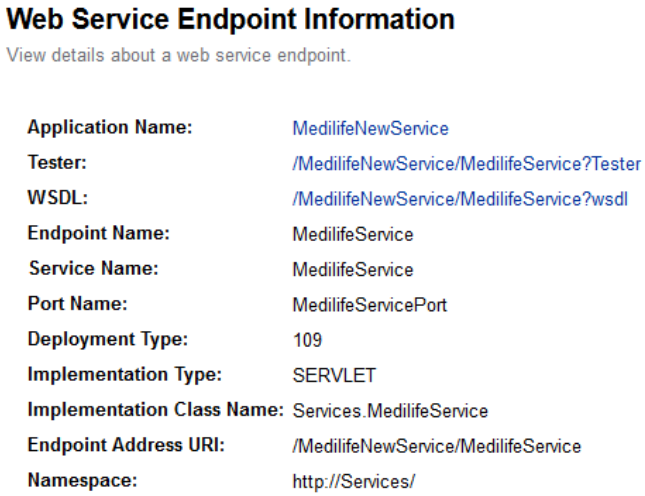
At this chapter, we applied the built quality model at the point of integration of selected systems and got results.

4.2 Apply the Model on System

4.2.1 Reusability & Usability

4.2.1.1 Existence of meta-information (EMI)

The web service supports us by information such as Name, Port, and URL… automatically without any additional work. In addition, we have WSDL that contains more information about function and parameter (number, received, return and Data Type). Figure 4.1 illustrate the result of built web service generated automatically.



Web Service Endpoint Information

View details about a web service endpoint.

Application Name:	MedilifeNewService
Tester:	/MedilifeNewService/MedilifeService?Tester
WSDL:	/MedilifeNewService/MedilifeService?wsdl
Endpoint Name:	MedilifeService
Service Name:	MedilifeService
Port Name:	MedilifeServicePort
Deployment Type:	109
Implementation Type:	SERVLET
Implementation Class Name:	Services.MedilifeService
Endpoint Address URI:	/MedilifeNewService/MedilifeService
Namespace:	http://Services/

Figure (4.1) information about web services.

- As information mentioned previously, the **EMI** is **one**.

4.2.1.2 Self-Completeness of Service's Return Value (SCSr)

$$SCSr = \frac{\sum Mbr}{\sum Mb} = \frac{1}{4} = 0.25 = 25\%$$

4.2.1.3 Self-Completeness of Service's Parameter (SCSp)

$$SCSr = \frac{\sum Mbp}{\sum Mb} = \frac{1}{4} = 0.25 = 25\%$$

Note: we use the web service within enterprise. So, we do not focus on unknown parameter for other people. However, if we need to measure public web service that is available for all people then it will be inefficient.

4.2.1.4 Result

After measuring the reusability and usability, it came out during the use of SOA on integration, the reusability was high and the usability was normal.

4.2.2 Reliability

4.2.2.1 Mean Time to Repair (MTTR)

Mean Time to Repair = (Total down time) / (number of breakdowns).

All system failure according to the database the **MTTR = 48 h / 2 = 24 h/break.** But if we are not taking the database failure in consideration the **MTTR = zero h/break.**

4.2.2.2 Mean Time Between Failures (MTBF)

MTBF = summations of time between failures / total number of failures

All system failure according to the database the **MTBF = 33 d / 2 = 16.5 day/failure.** But if we are not taking the database failure in consideration the **MTTR = 0 h/break.**

4.2.2.3 Result

After measuring the **Reliability,** it came out during the use of SOA on integration, the Reliability was high, and all problems that were not related to the integration were eliminated.

4.2.3 Performance

Table 4.1 explain the experiments of measure the time of send the request, time of receive the response and the different between two time on ten different experiments.

Table (4.1) illustrates the experiments results

#	Time Request	Time Response	Different Between Two Time
1	1119	1587	468
2	3309	3777	468
3	9501	9969	468
4	2670	2826	156
5	9311	9779	468
6	1999	2311	312
7	3957	4425	468
8	2940	3252	312
9	4096	4564	468
10	1949	2417	468

Notes: Time measured by millisecond.

4.2.3.1 Service Response Time (SRT)

STR = (468+468+468+156+468+312+468+312+468+468)/10 = **405.6 milliseconds**.

4.2.3.2 Throughput of Service (TPSRV)

From the previous table we can calculate the **(TPSRV) = 2 Request/Second**.

4.2.3.3 Timeliness

The timeliness we accepted is **1 second**. According to table (4.1), the timeliness has been achieved.

4.2.3.4 Result

After measuring the performance, it came out during the use of SOA to integrate systems; the performance was not effected and still accepted.

4.2.4 Modifiability

4.2.4.1 Extensibility

In case of SOA, the characteristic of extensibility it is not a problem because of the web service independence, and the number of interfaces that can be added depends on our capability of equipment such as servers, network and so on.

 # Interface that can be added = ∞. Based on our capability and needs.

4.2.4.2 Testability

Some test cases are automatically created when we create web services.

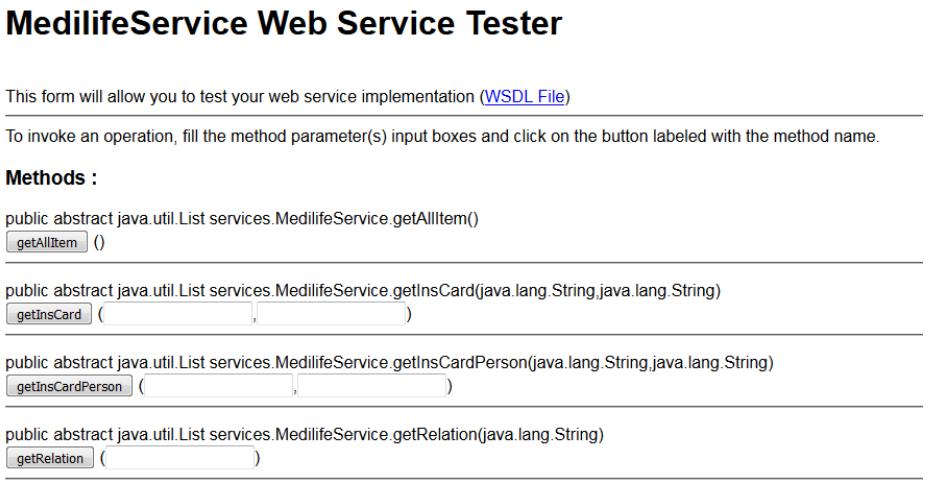


Figure (4.2) generated test methods.

#Test case per a method or function **(TC per M) = one test case/method.**

4.2.4.3 Result

Because of the nature of SOA, there is very high ability to extend function, equipment and so on. Moreover, it generates test case automatically then there is high Modifiability.

4.2.5 Correctness

4.2.5.1 Defects per KLOC - most common measure for correctness (Defects per KLOC)

The measure only applied on integration points (web services).

Defects per KLOC = zero defects.

4.2.5.2 Defects ware counted over a specified period (Defects per period)

Over two weeks there is **zero** defects causing failure (stopped service), except network problems. However, there is **one** function returned exception in one case.

4.2.5.3 Number of non-conformance service / total number of services (percentage of Non-CS)

All services are conformance to the specification then the **percentage of Non-CSs are 0%.**

4.2.5.4 Result

After measuring the Correctness, it came out during the use of SOA to integrate systems; the precision of correctness reached its best cases.

4.2.6 Results Summary

After applying quality model on integrated system, the results gained from the measuring process were high rate for the reusability, correctness, reliability and extensibility. Moreover, normal rate for the testability, usability and performance. Knowing that the extensibility is high and the testability is normal, regarding to the fact that both of them are sub-attribute of modifiability, the worst case will be chosen; which is normal. Table 4.2 show all results.

Table (4.2) Show the results of implementation.

Attributes	Metrics	Result
Reusability & Usability	Existence of meta-information	**One** (Zero or One)
	Self-Completeness of Service's Return Value	**25%**
	Self-Completeness of Service's Parameter	**25%**
Reliability	Mean Time to Repair	**Zero** hour/break.
	Mean Time Between Failures	**Zero** day/failure
Performance	Service Response Time	**405.6** milliseconds.
	Throughput of Service	**2** Request/Second.
	Timeliness	**405** milliseconds (accepted Time is one second).
Modifiability	Extensibility	∞ infinite
	Testability	**One** test case/method
Correctness	Defects per KLOC	**zero** defects per codes
	Defects per period	**Zero** per 2 weeks
	percentage of Non-CS	**0%**

Chapter Five

Conclusions and Recommendations

5.1 Conclusions

The result of this research is a quality model dedicated to evaluate the systems after being integrated by the SOA technique. The model includes a set of quality attributes selected based on the concepts of integration and SOA. The metrics used to measure these attributes were then determined by applying this model to the final system by linking two pre-selected systems and integrate them by building Web service as an implementation for SOA technology.

5.2 Recommendation

As a complement to this Study, there are some recommendations for researchers in this subject to improve the quality model:

- Add a security attribute to the quality model. Because security is a large concept and it needs intensive research than the rest of the other attributes and cannot be taken in short or partial way.
- Extend the metrics that used to measure the attributes of the model.
- Apply the model in different similar cases or systems (using same techniques) and compare the results.

6. Reference

1. Balfagih, Zain, and Mohd Fadzil Hassan. "Quality model for web services from multi-stakeholders' perspective." Information Management and Engineering, 2009. ICIME'09. International Conference on. IEEE, 2009.
2. Deissenboeck, Florian, et al. "Software quality models: Purposes, usage scenarios and requirements." Software Quality, 2009. WOSQ'09. ICSE Workshop on. IEEE, 2009.
3. Sage, Andrew P., and William B. Rouse, eds. Handbook of systems engineering and management. John Wiley & Sons, 2009.
4. Land, Rikard, and Ivica Crnkovic. "Existing approaches to software integration and a challenge for the future." integration 40 (2004): 58-104.
5. IEEE, IEEE Standard Glossary of Software Engineering Terminology, report IEEE Std 610.12-1990, IEEE, 1990.
6. Losavio, Francisca, Dinarle Ortega, and María Pérez. "Modeling EAI [Enterprise Application Integration]." Computer Science Society, 2002. SCCC 2002. Proceedings. 22nd International Conference of the Chilean. IEEE, 2002.
7. Lin, Goh Chun, et al. "A Fresh Graduate's Guide to Software Development Tools and Technologies." *National University of Singapore* (2011).
8. Munassar, Nabil Mohammed Ali, and A. Govardhan. "A comparison between five models of software engineering." IJCSI 5 (2010): 95-101.
9. Sommerville Ian Software Engineering 8 Edition Pearson Education 2007
10. Milicic, Drazen. "Software quality models and philosophies." Software quality attributes and trade-offs (2005): 3-19.
11. Khosravi, Khashayar, and Yann-Gaël Guéhéneuc. "A quality model for design patterns." University of Montreal, Tech. Rep (2004).
12. Farooq, Sheikh Umar, S. M. K. Quadri, and Nesar Ahmad. "Software measurements and metrics: Role in effective software testing." International

Journal of Engineering Science and Technology (IJEST) 3.1 (2011): 671-680.

13. Clark, Kim J. "Integration architecture: Comparing web APIs with service-oriented architecture and enterprise application integration." (2015).
14. Choi, Si Won, and Soo Dong Kim. "A quality model for evaluating reusability of services in soa." E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on. IEEE, 2008.
15. Lee, Jae Yoo, Jung Woo Lee, and Soo Dong Kim. "A quality model for evaluating software-as-a-service in cloud computing." Software Engineering Research, Management and Applications, 2009. SERA'09. 7th ACIS International Conference on. IEEE, 2009.
16. Goeb, Andreas, and Klaus Lochmann. "A software quality model for SOA." Proceedings of the 8th international workshop on Software quality. ACM, 2011.
17. Khan, Imran Akhtar, and Roopa Singh. "Quality assurance and integration testing aspects in web based applications." arXiv preprint arXiv:1207.3213 (2012).
18. Dehmi Kalan, Jaipur and Dehmi Kalan, Jaipur “A survey on Software as a service (SaaS) using quality model in cloud computing" January 2014 Page No. 3598-3602 (2014).
19. Rawashdeh, Adnan, and Bassem Matalkah. "A new software quality model for evaluating COTS components." Journal of Computer Science 2.4 (2006): 373-381.
20. Choi, Si Won, Jin Sun Her, and Soo Dong Kim. "Modeling QoS attributes and metrics for evaluating services in SOA considering consumers' perspective as the first class requirement." Asia-Pacific Service Computing Conference, the 2nd IEEE. IEEE, 2007.

21. How to calculate Software Quality Attributes, 20-7-2017 12:34 PM, http//www.qasigma.com/2008/12/how-to-calculate-software-quality-attributes.html.
22. Ahenkan, Nana. ASSESSING THE IMPORTANCE OF QUALITY ATTRIBUTES AND METRICS IN MOBILE GEOGRAPHICAL INFORMATION SYSTEMS (GIS) APPLICATIONS. MS thesis. 2010.