SUDAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF POSTGRADUATE STUDIES

A Customized Quality Model To Evaluate Domain Engineering In Software Product Line

نموذج جودة مخصص لتقييم هندسة المجال في خط إنتاج البرمجيات

A Thesis Submitted In Partial Fulfillment Of The Requirements Of Master Degree In Computer Science

PREPARED BY

Lemia Alamin Algmri Mohammed

SUPERVISED BY

Dr. Abd El hamid Abd El hadi Mansor

March 2019

ألاية

قَالَ اللهُ تَعَالَىٰ:

﴿ وَلَقَدْ كَرَّمْنَا بَنِيٓ ءَادَمَ وَحَمَلْنَٰهُمْ فِي ٱلْبَرِّ وَٱلْبَحْرِ وَرَزَقْنَٰهُم مِّنَ ٱلطَّيِّبَٰتِ وَفَضَّلْنَٰهُمْ عَلَىٰ كَثِيرٍ مِّمَّنْ خَلَقْنَا تَفْضِيلًا ﴾

سورة الإسراء الآية رقم(70)

Dedication

To my parents, Alamin Algmri and Aisha Taha who supported me to continue in my educational process, my colleagues and my teachers who provided me with advice that helped me in my thesis.

Abstract

software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way , its recognized as a successful approach to reuse in software development .

Software quality models have been widely used for assessing software quality during system development .howevertheir use for evaluating the outputs of software product line phases,such as the domain engineering outputs (Family Architecture and the core assets implementation) has been extensively not reported.

This research proposes a quality model to evaluate the variability of the common parts (domain engineering outputs) which are very importance for each product in the family. Moreover, the research evaluates the previous research in software product line carefully to identify a proper way to develop the proposed model. The proposed model is customized from (ISO/IEC 9126-1) model, in addition to some related concepts extracted from the key features of software product line development approach. Furthermore, the model will evaluate the core assets at early development phases in order to accurate prediction of final system quality, hence producing products in an effective way(lower cost and shorter time).

The variability of core assets and product line architecture is evaluated based on the functionality, reliability, usability and efficiency measurements .each attribute has its weight in the whole variability value.

The proposed model applied for the core assets and general structure of the administrative package of Sudan university of science and technology and the result of evaluation s that the variability of employees and user modules(which are the common modules in the package) is 77.89% .

المستخلص

تم تعريف خط انتاج البرمجيات على أنه مجموعة من الأنظمة كثيفة البرامج التي تشترك في مجموعة مشتركة من الميزات التي تلبي الاحتياجات المحددة لقطاع ما سوق معين والتي يتم تطويرها من مجموعة مشتركة من الأصول الأساسية بطريقة محددة ، حيث تعتبر هذه المنهجية طريقة معترف بها و ناجحة لتحقيق إعادة الاستخدام في تطوير البرمجيات.
تم استخدام نماذج جودة البرمجيات على نطاق واسع لتقييم جودة المنتجات البرمجية أثناء تطوير النظام . و مع ذلك استخدامها لتقييم مخرجات مراحل خط إنتاج البرمجيات ، مثل مخرجات مرحلة هندسة المجال (معمارية العائلة البرمجية وتطبيق الأصول الأساسية) لم يكن على نطاق واسع.
يدرس هذا البحث الأبحاث الحالية في خط إنتاج البرمجيات بعناية ويجد حلاً للمشكلة , حيث يتمثل هذا الحل في تصميم نموذج جودة لتقييم تباين الأجزاء المشتركة (مخرجات مرحلة هندسة المجال). الأجزاء المشتركة ذات أهمية كبيرة لكل منتج في خط الانتاج . النموذج المقترح تم تخصيصه من (ISO / IEC 9126-1) ، بالإضافة إلى بعض المفاهيم ذات الصلة المستخرجة من السمات الرئيسية لنهج تطوير خط إنتاج البرامج. سيقوم النموذج المقترح بتقييم الأصول الأساسية في مراحل التطوير المبكرة من أجل التنبؤ الدقيق بجودة النظام النهائية ، وبالتالي إنتاج انظمة بتكلفة أقل وفي وقت أقصر.
يتم تقييم تباين الأصول الأساسية ومعمارية خط الإنتاج بناءً على خصائص الوظيفية والموثوقية وقابلية الاستخدام وقياس الكفاءة. كل خاصية من هذه الخصائص لها وزنها في قيمة التباين الكلية .
تم تطبيق النموذج المقترح على الأصول الأساسية والهيكل العام لحزمة الانظمة الإدارية لجامعة السودان للعلوم والتكنولوجيا ,حيث اظهرت نتائج التقييم أن وحدتي الموظفين و المستخدمين (والتي هي الوحدات المشتركة في الحزمة) نسبة التباين فيهما هي 77.89 ٪ .

Table Of Content

List of figure

List of tables

Abbreviations

Abbreviation	Terms
SPL	Software Product Line
QA	Quality Assurance
DE	Domain Engineering
AE	Application Engineering
FIC	Functional Implementation Completeness
ISC	Interface Stander Compliance
MTBF	Mean Time Between Failure
Un.I/O	Understandability Of Input And Output
Thr	Throughput
MTTR	Mean Time To Response
Cl.Dep	Classes Dependency
Str.Doc	Strength Of Document
TV	Total Variability
PLA	Product Line Architecture

Chapter I
Introduction

Chapter I

Introduction

1.1 Background

Software product families are recognized as a successful approach to reuse in software development (M. Jazayeri, A. Ran, F. van der Linden , 2000) . The philosophy behind software product families is to economically exploit the commonalities between software products, but at the same time preserve the ability to vary the functionality between these products. Managing these differences between products, referred to as variability, is a key success factor in product families . Research in the context of software product families is shifting from focusing on exploiting the commonalities towards managing the variability, referred to as variability management, e.g. (M. Clauss , 2001). A key aspect of variability management is the explicit representation of the variability.

Previous researches like : (M. Clauss , 2001) ,(F. Bachman, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, A. Vilbig , 2003) and (S. Thiel, A. Hein , 2002) agrees that variability should be modeled uniformly over the lifecycle, and in terms of dependencies and first class represented variation points. Educational case studies we performed at organizations that employ medium and large-scale software product families have shown that, in addition to providing the required functionality, the main focus during product derivation is on satisfying complex dependencies, i.e. dependencies that affect the binding of a large number of variation points, such as quality attributes.

In the past few years, several approaches have been developed for modeling the variability in software product families (F. Bachman, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, A. Vilbig , 2003), (M. Clauss , 2001) and (R. van Ommering , 2000) . Most of these approaches treat variation points as first-class citizens and provide means to model simple dependencies, and some of the approaches model the variability uniformly over the lifecycle. None of the approaches, however, supports the modeling of complex dependencies.

this research present variability model that measure the variability in all abstraction layers of the software product family. It treats variation points as points collected from other related characteristics and provides means to model the relations between (functionality ,usability ,maintainability , efficiency ,reliability). The proposed solution is applied with a management case study .

1.2 Research Problem

The basic problem in SPL development approach is that there is no sufficient applicable quality model to evaluate domain engineering phase output in software product lines developing techniques.Therefore, we need quality models that are flexible and reusable (Adam Trendowicz&Teade Punter ,2018) .
Currently, according to our best knowledge,we concluded that no single quality model for assessing all our requirements (domain design and domain implementation phases).

1.3 Research Questions

There are many quality approaches to evaluate the quality of the phases of SPL in the domainofengineering and application engineering, but the main question of this research is:

❖ How to evaluate (to make sure that) the common components of the product family are variable and suitable for all the domain products beforebeginning the application development?

1.4 Research Objectives

The main objectives of this research are:

- To evaluate the current research in software product line
- To propose a new model to evaluate the quality of products in software product lines.
- To evaluate the proposed model.

1.5 Research Significance

The proposed model in this research will evaluatethe structure and core assets of software product lines product, which is difficult to evaluate using the other models of quality assurance.
The proposed model will evaluate the core assets at early development phases in order to give an accurate prediction of final system quality, hence producing software with lower costs and in a shorter time .

1.6 Research Methodology

The methodology of this research is based on studding the problem domain requirements , the benefit of the SPL approach ,the founded solutions ,define a new solution and prove the applicability of the proposed solution .

1.7 Research Scope

The proposed model will evaluate the quality product line Architecture (PLA) in SPL products which means the abstract level of the devolopment process as shown in figure(1-1).

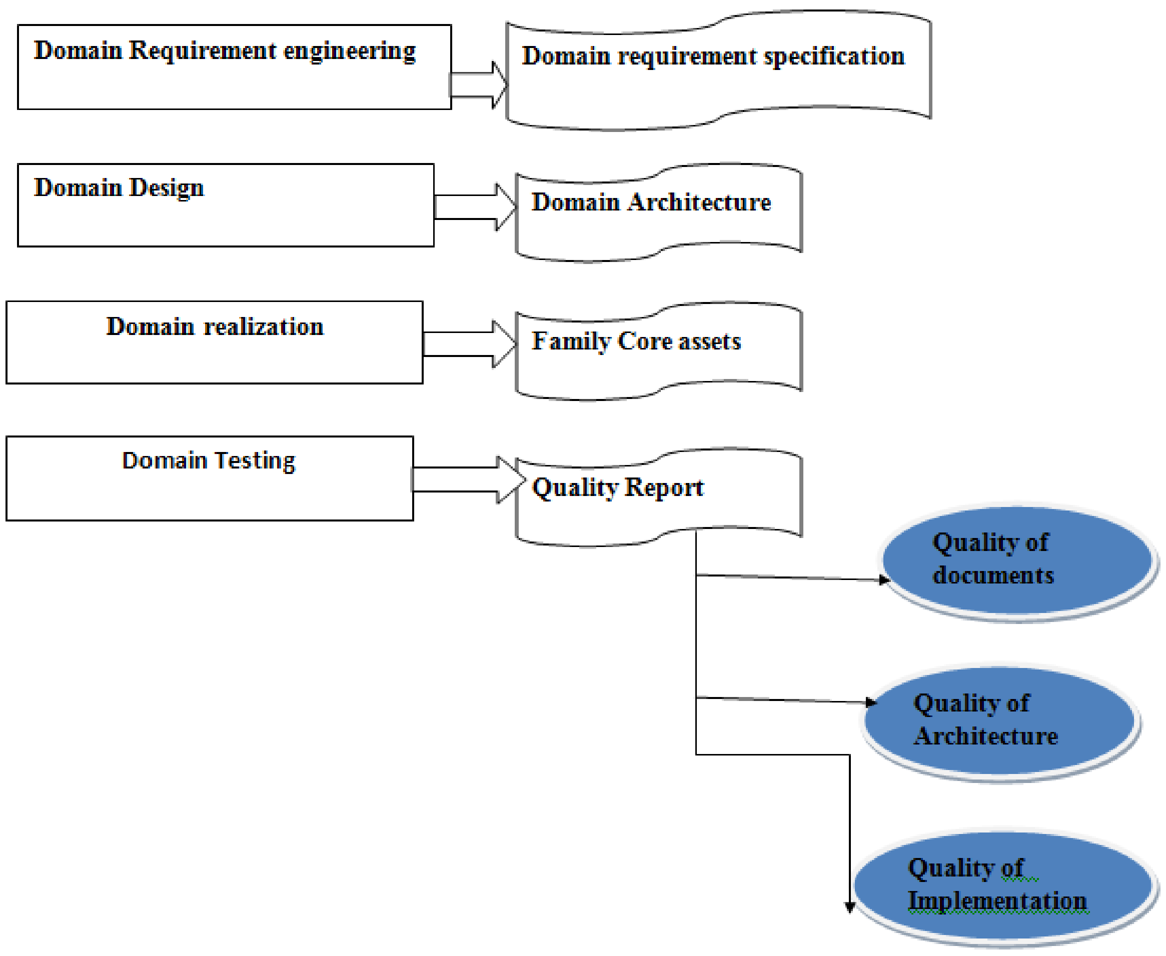


Figure (1-1) The domain of the research

1.8 Research Layout

Chapter One Contain introduction about the project, defining the problem, objectives, methodology, and scope.
Chapter Two Contain a general background in software quality assurance, software product lines, quality models and related works.
Chapter Three Contain the proposed model.
Chapter Four Contain the Metrics and Evaluation method.
Chapter Five Contain the Evaluation and result discussion.
Chapter Six Contain the conclusion and recommendations.

Chapter Tow
Literature Review

CHAPTER Two

Literature Review

Overview:

This chapter, contain some review for the basic concepts about Software Quality assurance,quality models, the Software product Lines, metrics concept. And present some related works.

2.1 Software Quality Assurance

Is defined as Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software .The goal of quality assurance is to provide managers with the necessary data to be informed about product quality by gaining insight and confidence that the product is meeting its goals, if the data provided through quality assurance identify problems, it is management’s responsibility to address the problems and apply the necessary resources to resolve quality issues.

The term quality involves two aspects which are quality of design, and quality of conformance and the good quality will be obtained if both of them are controlled satisfactorily:

2.1.1 Quality Of Design

Is the quality which the producer or supplier is intending to offer to the customer. When the producer is making the quality of design of the product, he should take into consideration the customer's requirements in order to satisfy them with **fitness for use** of the product.

2.1.2 Quality Of Conformance

Is the level of the quality of the product actually produced and delivered through the production or service process of the organization as per the specifications or design? When the quality of a product entirely conforms to the specification (design), the quality of conformance is deemed excellent.

2.2 Quality Model

A Quality Model is defined as, the set of characteristics (attributes) and the relationships between them which provides the basis for specifying quality requirements and evaluating quality .Software quality is described by quality models. there are three types of quality models firstly the general models such as McCall’s quality model (Mc

Call, J. A. &, Richards, P. K. & Walters, G. F,1977), Boehm model (, B. W., Brown, H., Lipow,M ,1978), FURBS quality models (Etxeberria, L., Sagardui, G. &Belategi, L., 2008), ISO/IEC 9126(Al-Rawashdeh, T.A., Al'Azzeh, F.M. & Al-Qatawneh, S.M., 2014).Secondly there are the tailored model that is specified for especial type or domain of products like GEQUAMO model (GEORGIADOU, E.L.L.I., 2003) , Alvaro model (Alvaro, A., Almeida, E.S.D. &Meira, S.L., 2007) and Rawashdeh model (Rawashdeh, A. &Matalkah, B., 2006).
and finally there are the open source quality models(Umm-E-Laila et al., 2017), (Adewumi, A., Misra , S. &Omoregbe, N., 2013) like: SQO-OSS Model (Samoladas, I. et al., 2008) and QualOSS Model (Glott, R. et al., 2010).

2.3 Software Product Lines (SPL)

Carnegie Mellon **Software** Engineering Institute defines a **software product line** as "a set of **software**-intensive systems that share a common, managed a set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."(Clements, P. & Northrop, L., 2012).
The characteristic that distinguishes software product lines from previous efforts is predictive versus opportunistic software reuse. Rather than put general software components into a library in the hope that opportunities for reuse will arise, software product lines only call for software artifacts to be created when reuse is predicted in one or more products in a well-defined product line.
SPL development process contains two steps: the **domain engineering** and **the application engineering**, the Domain Engineering phase contains four phases:

- ❖ **Domain Requirements Engineering** is the process of creating and managing requirements for the entire product line. A measure is therefore classified in this phase if it is defined for measuring the quality of domain requirements artifacts.

- ❖ **Domain Design** is the process of creating a common architecture for the SPL. A measure is classified in this phase if it is defined for assessing the quality of software artifacts related to the design of the whole product line and/or common architecture, taking into account the variability, the common parts, etc.

- ❖ **Domain Realization** is the process of creating the core assets. A measure is classified in this phase if it is defined to evaluate the quality of core assets.

- **Domain Testing** is the process of testing the common architecture and core assets created. A measure is therefore classified in this phase if it measures a quality attribute related to the testing phase (e.g., the quality of the proven cases).

2.3.1 Quality-aware Domain Engineering

Software product line practice wants to achieve a number of goals including reduced costs, improved time to market, and improved quality of the products belonging to the product line. These goals will only be achieved if quality attributes, such as correctness and reliability, are continuous objectives from the earliest phases of development(John McGregor, 2001).

The importance that quality and quality assurance is acquiring in reuse contexts is justified because an error in a reusable asset can be propagated to a lot of products. Regarding the previous classification of quality requirements, to date, *product-line quality attributes* such as extensibility, modifiability, etc. has received most of the attention: how to know if the line covers all the envisioned functionality of the products in the scope. Nevertheless, *domain-relevant quality attributes* have beenignored, especially the variability that those attribute can have in a software product line (Etxeberria, L., Sagardui, G. &Belategi, L., 2008).

There is a list of tasks or practices to perform during domain engineering to facilitate quality aware product line engineering as shown in Figure (1).

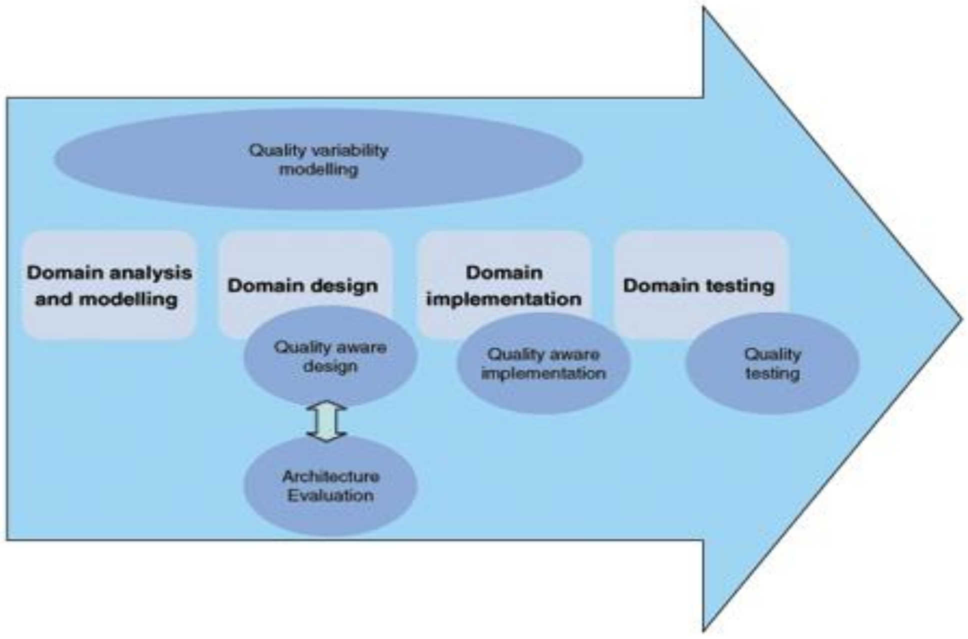


Figure (2-1) Qualities Aware Domain Engineering

2.4 Quality Metrics

A metric is a quantifiable measurement of software product, process, or project that directly observed, calculated, or predicted (Farooq , S.U., K, S.M. &Nesar Ahmad, 2014).

The main objectives of software quality metricsare to facilitate management control as well as the planning and execution of the appropriate managerial interventions. achievement of this objective based on a calculation of metrics regarding:

- ❖ 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 the Accumulation of metrics information regarding the performance of teams and units.

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 a numerical value from metrics. Then, each quality sub-factors assessed using their criteria. Finally, numerical value assigned to quality characteristics from their quality subfactors (Farooq , S.U., K, S.M. &Nesar Ahmad, 2014).See figure (2-2).

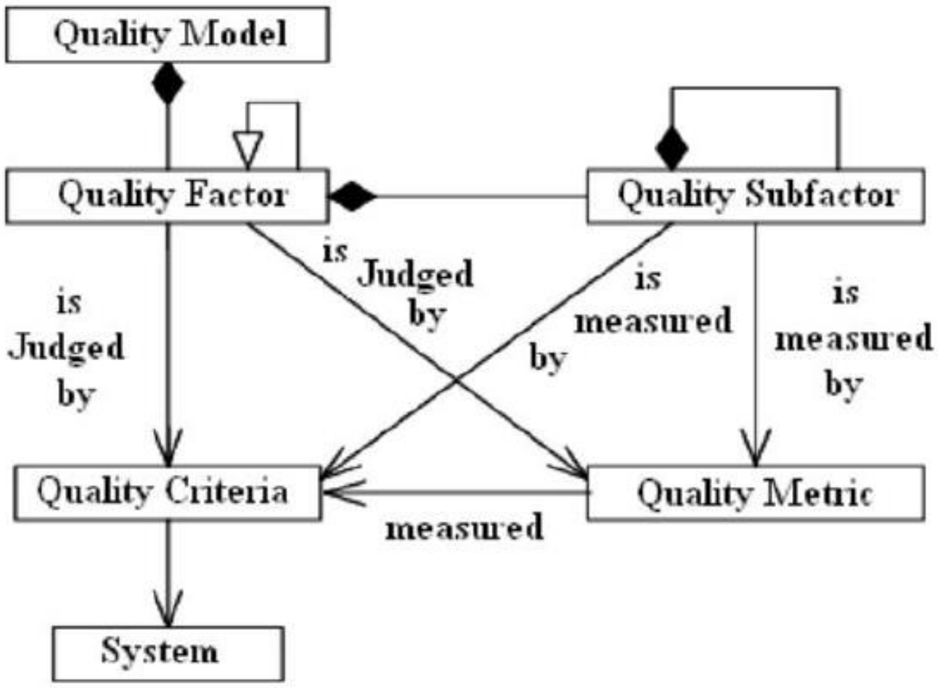


Figure (2-2)Relationship Among Quality Model Elements(Farooq , S.U., K, S.M. &Nesar Ahmad, 2014).

2.5 Related Works

2.5.1 Quality Modeling for Software Product Lines

This paper, investigate to which extent existing quality modeling approaches facilitate high-quality software product lines. Firstly, define several requirements for an appropriate quality model. Then, use those requirements to review the existing quality modeling approaches.

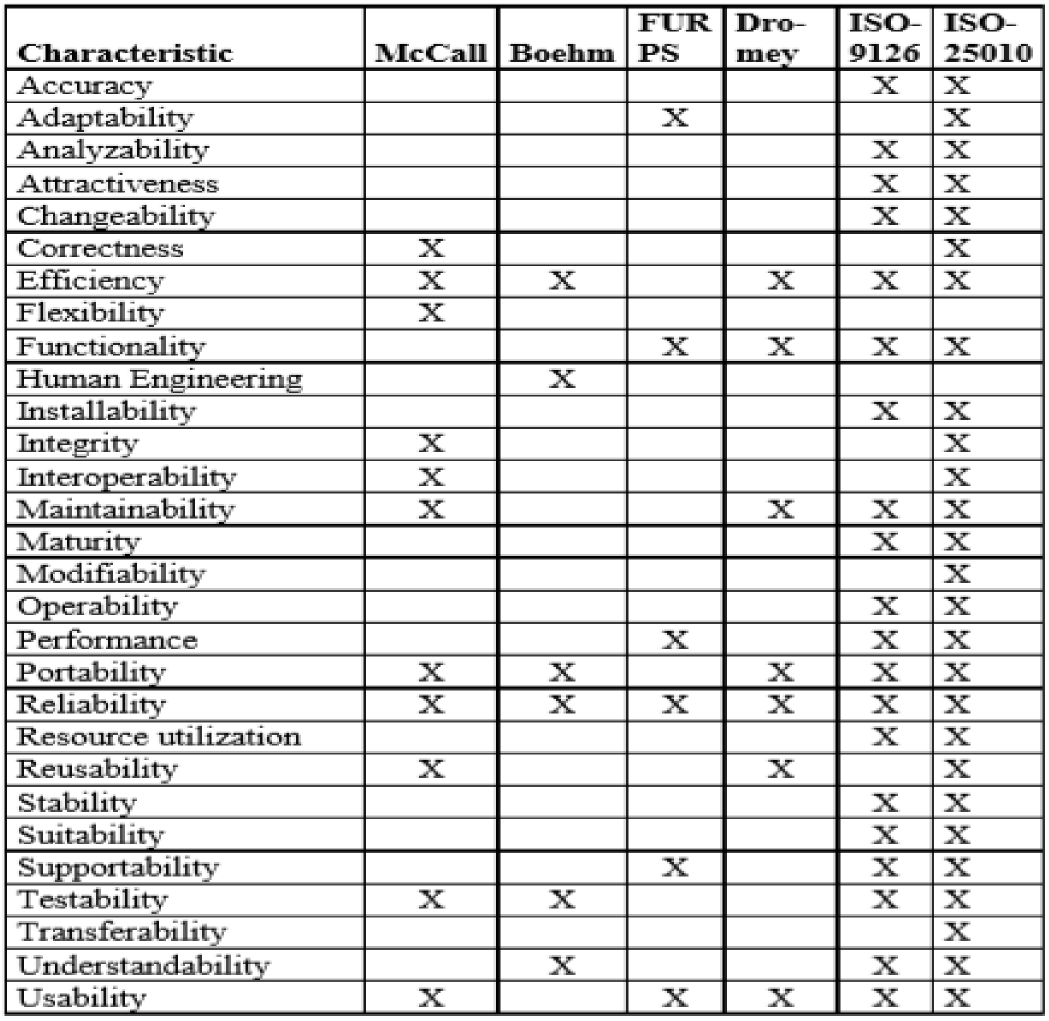
The conclusion from this review is that there is no single quality model fulfills all of our requirements. However, several approaches contain valuable characteristics. Based upon those characteristics, they propose the Prometheus approach,Which is a goal-oriented method that integrates quantitative and qualitative approaches to quality control.

This method starts quality modeling early in the software lifecycle and is reusable across product lines (Adam Trendowicz&Teade Punter, 2018).

2.5.2 A Review of Software Quality Models for the Evaluation of Software Products

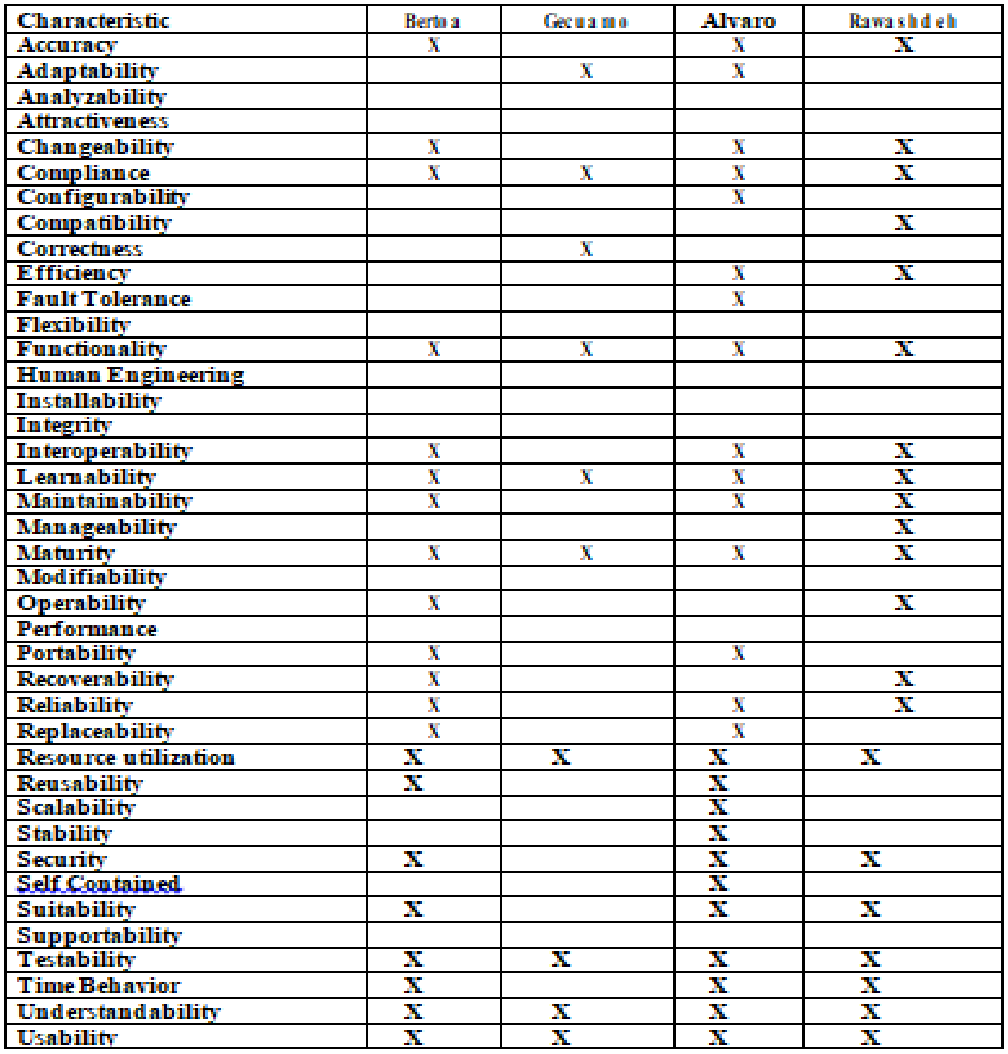
This study divided the quality models to two categories (open source quality models, basic quality model), comparing the coverage of the characteristics in the two types as is shown in the table (2-1) and table (2-2) (Miguel, J.P., Mauricio, D. & Rodríguez, G., 2014) .

Table (2-1)Comparison of Basic Models (Miguel, J.P., Mauricio, D. & Rodríguez, G., 2014)



Characteristic	McCall	Boehm	FUR PS	Dro-mey	ISO-9126	ISO-25010
Accuracy					X	X
Adaptability			X			X
Analyzability					X	X
Attractiveness					X	X
Changeability					X	X
Correctness	X					X
Efficiency	X	X		X	X	X
Flexibility	X					
Functionality			X	X	X	X
Human Engineering		X				
Installability					X	X
Integrity	X					X
Interoperability	X					X
Maintainability	X			X	X	X
Maturity					X	X
Modifiability						X
Operability					X	X
Performance			X		X	X
Portability	X	X		X	X	X
Reliability	X	X	X	X	X	X
Resource utilization					X	X
Reusability	X			X		X
Stability					X	X
Suitability					X	X
Supportability			X		X	X
Testability	X	X			X	X
Transferability						X
Understandability		X			X	X
Usability	X		X	X	X	X

Table (2-2) Comparison between tailored models (Miguel, J.P., Mauricio, D. & Rodríguez, G., 2014)



Characteristic	Bertoa	Georgiadou	Alvaro	Rawashdeh
Accuracy	X		X	X
Adaptability		X	X	
Analyzability				
Attractiveness				
Changeability	X		X	X
Compliance	X	X	X	X
Configurability			X	
Compatibility				X
Correctness		X		
Efficiency			X	X
Fault Tolerance			X	
Flexibility				
Functionality	X	X	X	X
Human Engineering				
Installability				
Integrity				
Interoperability	X		X	X
Learnability	X	X	X	X
Maintainability	X		X	X
Manageability				X
Maturity	X	X	X	X
Modifiability				
Operability	X			X
Performance				
Portability	X		X	
Recoverability	X			X
Reliability	X		X	X
Replaceability	X		X	
Resource utilization	X	X	X	X
Reusability	X		X	
Scalability			X	
Stability			X	
Security	X		X	X
Self Contained			X	
Suitability	X		X	X
Supportability				
Testability	X	X	X	X
Time Behavior	X		X	X
Understandability	X	X	X	X
Usability	X	X	X	X

2.5.3 2.5.3 A systematic review of quality attributes and measures for software product lines

this paper review all the available studies from 1996 to 2010 that present quality attributes and/or measures for SPL.

These attributes and measures have been classified using a set of criteria that includes the life cycle phase in which the measures are applied; the corresponding quality characteristics; their support for specific SPL characteristics, the procedure used to validate the measures, etc. the study found that 165 measures related to 97 different

quality attributes. The results of the review indicated that 92% of the measures evaluate attributes that are related to maintainability. In addition, 67% of the measures are used during the design phase of Domain Engineering, and 56% are applied to evaluate the product line architecture. However, only 25% of them have been empirically validated.
In conclusion, the results provide a global vision of the state of the research within this area in order to help researchers in detecting weaknesses, directing research efforts, and identifying new research lines. In particular, there is a need for new measures with which to evaluate both the quality of the artifacts produced during the entire SPL life cycle and other quality characteristics. There is also a need for more validation (both theoretical and empirical) of existing measures (Montagud, S., Abrahão, S. &Insfran, E., 2011).

2.5.4 A Bibliometric Analysis Of 20 Years Of Research On Software Product Lines

This paper analyzes the literature on product lines from 1995 to 2014, identifying the most influential publications, the most researched topics, and how the interest in those topics has evolved along the way. Bibliographic data have been gathered from the ISI Web of Science and Scopus. The data have been examined using two prominent bibliometric approaches: science mapping and performance analysis.
The result is that, software architecture was the initial motor of research in SPL, work on systematic software reuse has been essential for the development of the area and feature modeling has been the most important topic for the last 50 years, having the best evolution behavior in terms of a number of published papers and received citations .the evolution of the interest in those topics and the relationships among topics. Performance analysis has been used to recognize the most influential papers, the journals ,and conferences that have published (Heradio, R. et al., 2016).

2.5.5 Software Product Line Engineering And Variability Management: Achievements And Challenges

This study identified over 600 relevant research and experience papers published within the last seven years in established conferences and journals. and briefly summarize the major research achievements of these past seven years. The structure of this research summary along with a standardized software product line framework.
As a result, there are three highlighted trends that will have an impact on SPLE research in the next decade: firstly,managing variability in non-product-line settings, secondly, leveraging instantaneous feedback from big data and cloud computing during SPLE, and finally, addressing the open world assumption in software product line settings. Those

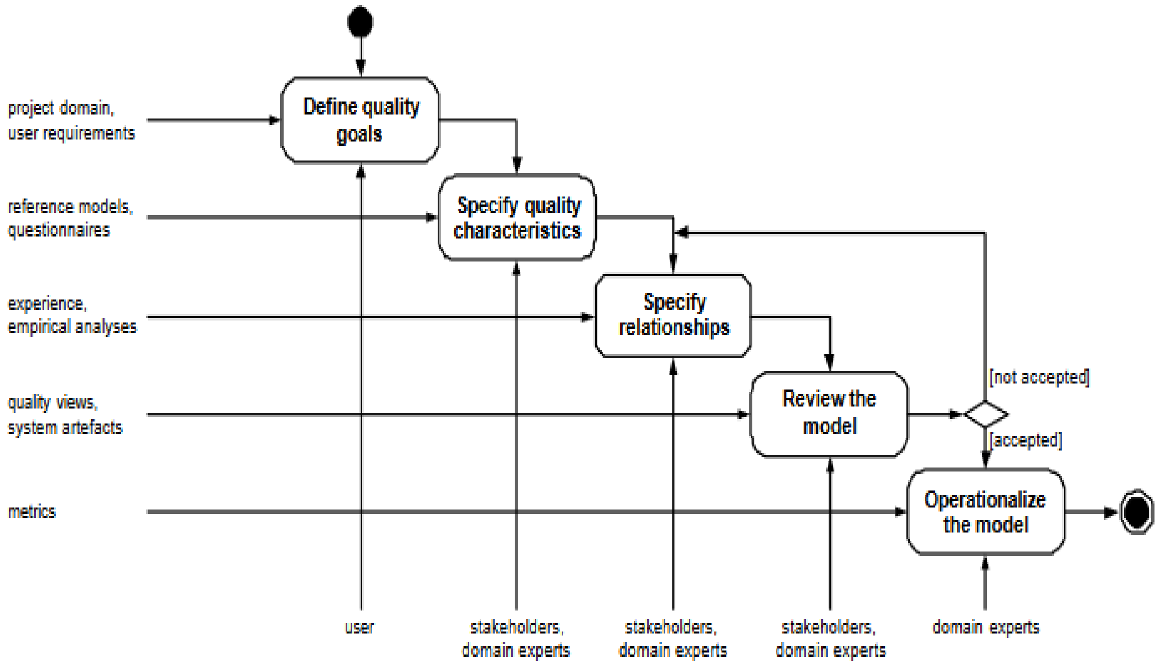
trends clearly indicate that research opportunities arise at the intersection between software product line engineering and service-oriented computing, cloud computing, big data analytics, autonomic computing ,and adaptive systems, to name the most important ones (Metzger, A. & Pohl, K., 2014).

2.6 Summary

The result of the whole overview is that there is a poorness of researches in the software product line quality modeling , al most of studies discuss the software quality in general and try to proposed approaches for evaluating software product line successful using the general models . Figure (2-2) discuss the process of creating a new quality model which will be the main part of methodology of this research .

Note

* The Quality models which were mentioned in this chapter are shown in the appendix .



project domain, user requirements
reference models, questionnaires
experience, empirical analyses
quality views, system artefacts
metrics
Define quality goals
Specify quality characteristics
Specify relationships
Review the model
Operationalize the model
[not accepted]
[accepted]
user
stakeholders, domain experts
stakeholders, domain experts
stakeholders, domain experts
domain experts

Figure (2-2) Activities during the specification phase of the Prometheus approach

Chapter Three
RESEARCH METHODOLOGY

Chapter Three

Research Methodology

This chapter presents the methodology to be adopted in continuing this research, Research procedures, software product line benefits and the schedule are included**.**

3.1 Research Design Procedure

Five main phases describe the research procedure: Identify Software Product Line Domain Engineering Benefits, Mapping Benefits to Quality Attribute, build the proposed model ,select Metrics for Each Quality Characteristics, apply the Model in the target system .see figure (3-1).

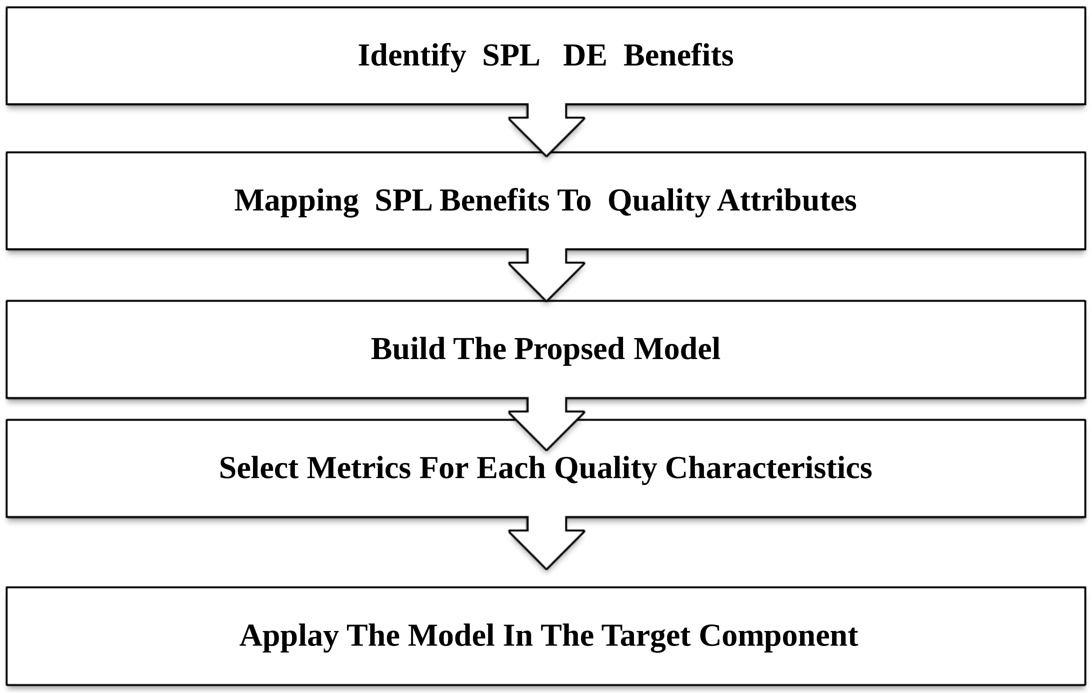

Identify SPL DE Benefits
Mapping SPL Benefits To Quality Attributes
Build The Propsed Model
Select Metrics For Each Quality Characteristics
Applay The Model In The Target Component

Figure (3-1) the methodology of this research

3.2 Identify SPL DE Benefits

To define a quality model for evaluating SPL, it is necessary to identify key features of SPL because the need of evaluation quality is coming from the benefits of the product.

The (SEI, 2016. Key Benefits) lists the following benefits associated with software product lines. as shown in figure (3).Large-scale productivity gains, Decreased time to market, Increased product quality, Decreased product risk, Increased market agility, Increased customer satisfaction, More efficient use of human resources, Ability to affect mass customization, Ability to maintain market presence, Ability to sustain unprecedented growth**.**

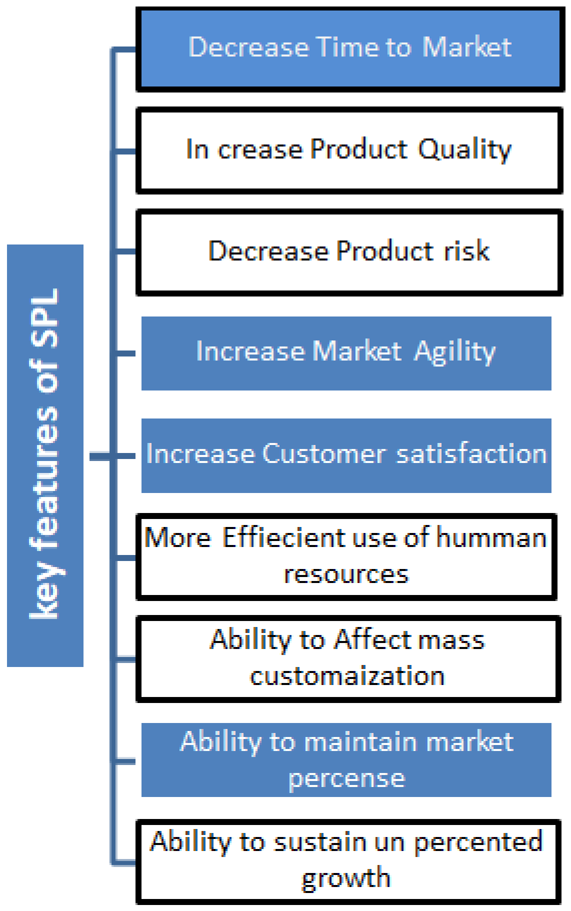

key features of SPL
Decrease Time to Market
In crease Product Quality
Decrease Product risk
Increase Market Agility
Increase Customer satisfaction
More Effiecient use of humman resources
Ability to Affect mass customaization
Ability to maintain market percense
Ability to sustain un percented growth

Figure (3-1) Benefits of SPL

3.3 Mapping SPL Benefits to Quality Attribute

Based on ISO/IEC 9126, Efficiency and Reliability are extended to cover specific features. Derived from the key features, maintainability, usability, and variability are newly defined.

Figure (3-3) shows a mapping relationship between the key features of and the quality attributes. Here we discuss for each attribute, its definition and our rationale reason for selecting it and in figure (3-4) we have all entities for the proposed model.

- ❖ Functionality :Functionality is the essential purpose of any product , so for each software we must assure that the final product meets the Software Requirement Specification (SRS) and get the user satisfaction , this characteristics divided into Sub_ characteristics: Suitability that is the essential Functionality characteristic and refers to the appropriateness (to specification) of the functions of the software. , correctness is the Degree to which a product or system provides the correct results with the needed degree of precision.

- ❖ Reliability: Reliability is about the Degree to which a component performs specified functions under specified conditions for a specified period of time. The rationale for defining this characteristic is that the SPL assets and Product Line Architecture (PLA) will be used in manyproducts, so it must be reliable before used and connected with other components, this characteristic is defined by Fault tolerance which is the Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

- ❖ Usability: Because the main purpose of the product is the user satisfaction, it's very important to get the user satisfaction by achieving usable functions, we check usability using the sub-characteristics understandability which determines the ease of which the functions of the system can be understood, relates to user mental models in Human-Computer Interaction methods.

- ❖ Efficiency: This characteristic is concerned with the system resources used when providing the required functionality. The amount of disk space, memory, etc.

  We divided it into Time behavior which is the Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements and Resource Behavior which Characterizes resources used like memory, CPU, disk and network usage.
- ❖ Maintainability: This characteristic representing the degree of effectiveness with which a product or system can be modified to improve it, correct it or adapt it to

changes in the environment, and in requirements. We select two characteristics for it.

They are Modularity which is the Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components, and Analyzability which is the Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

❖ Variability: oneofthe main differences between SPL developing approach and others is variability which means how the applications of a product line can differ. Together with the commonalities and defines the scope of a product line. We select the component variability sub_ characteristics for it which shows the ability of a component to operate in different systems with its same structure.

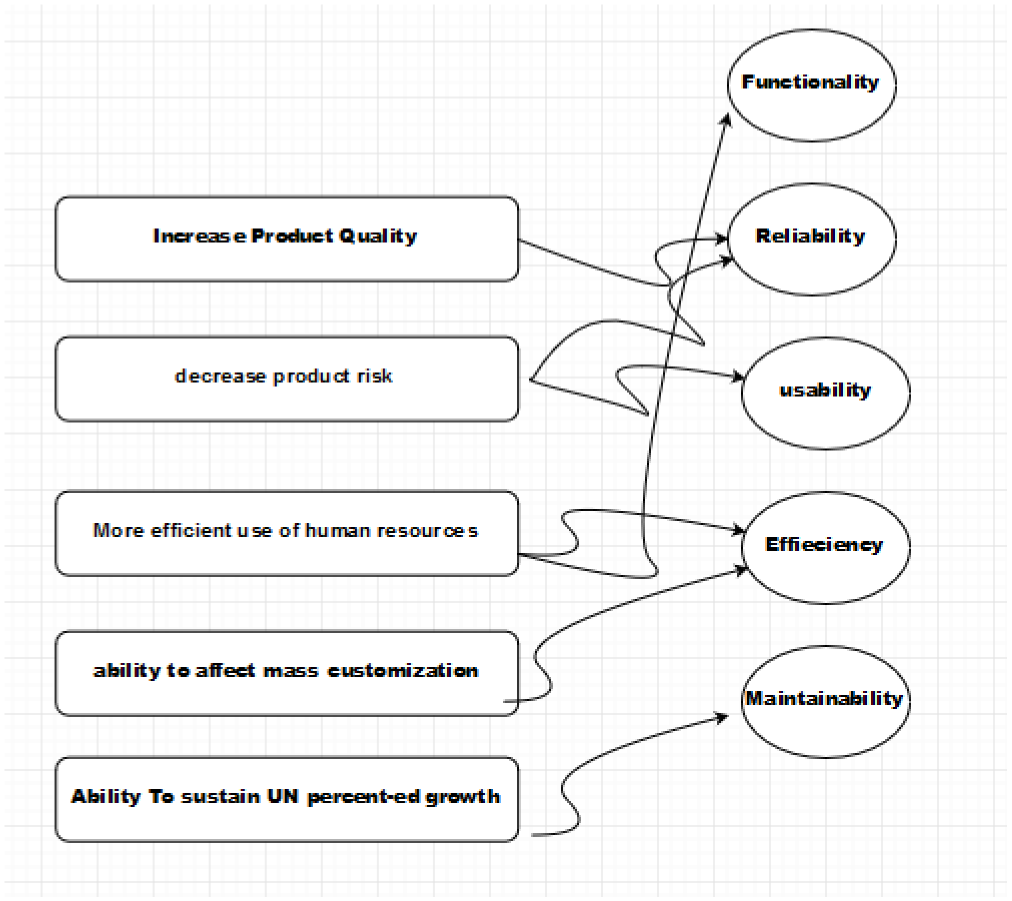

Functionality
Reliability
usability
Effieciency
Maintainability
Increase Product Quality
decrease product risk
More efficient use of human resources
ability to affect mass customization
Ability To sustain UN percent-ed growth

Figure (3-2) Mapping SPL Benefits to Quality Attributes

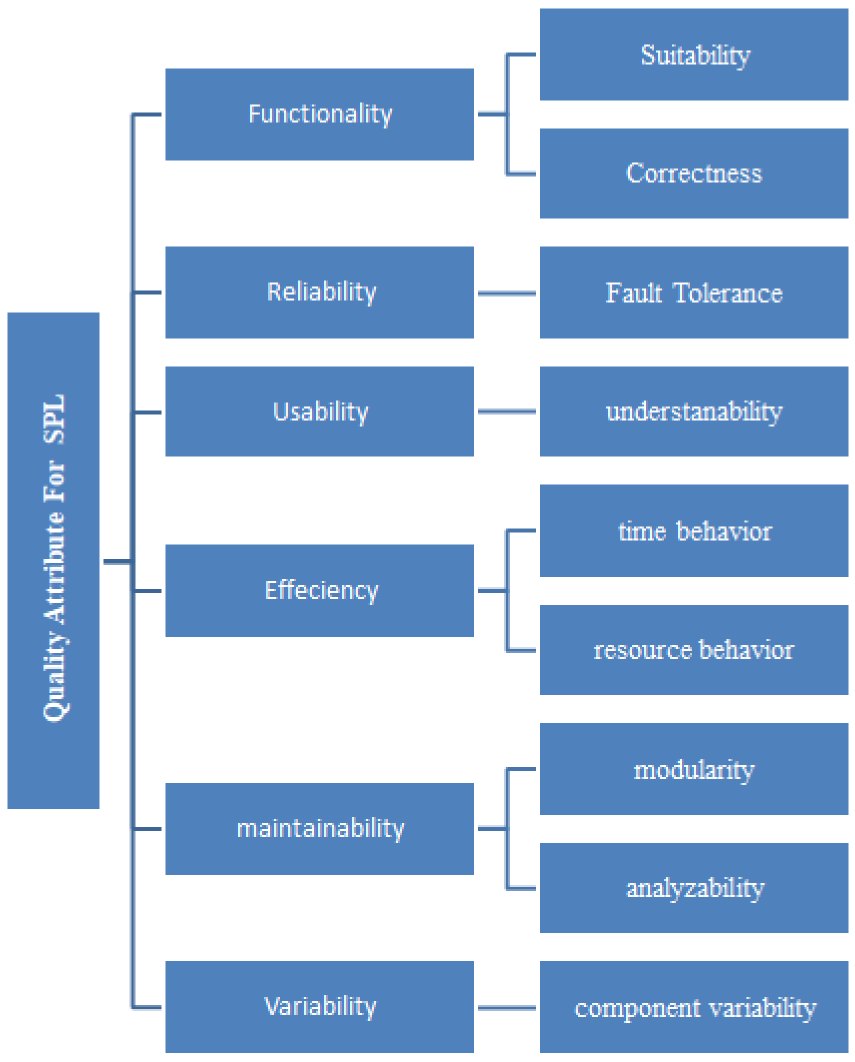


Figure (3-3) the Proposed Model

3.4 Case Study: Administrative Systems Package for Sudan University of Science and Technology:

The administrative systems package consists of 6 systems (3 of which work on them and the others are in the development phases). They are the Academics affairs system, the financial management system, the engineering management system, the legal administration system, the security, and safety system, the medical management system, the management and the services system .

All these systems are shared in the Programming method, screen design, and reporting formats.

The database of these systems is designed using Oracle 12c to design the database schema and the user interfaces are designed using CodeIgniter and bootstrap in the design of PHP user interfaces.

the system database consists of more than 100 tables, 13 of them are common among all systems and 36 common functions.

The validation will be divided into two sections, firstly the evaluation of the general structure using formal inspection for the database documents and secondly the evaluation of core assets (common functions) using the cognitive walkthrough approach.

CHAPTER FOUR EVALUATION METRICS

CHAPTER FOUR
EVALUATION METRICS

This chapter, discuss the selected metrics for each quality attribute and provide a description including formula, value range, relevant interpretations and the proposed model with the selected metrics as is shown in figure (4-1).

4.1 Functionality

4.1.1 Suitability

This attribute will be checked by the **functional Implementation Completeness (FIC) Metric** which measures the completeness of the implementation according to requirement specifications .The metrics will be computed with the following equation.

$$FIC = 1 - (A \div B)$$

$$\therefore A \equiv \textbf{Number of missing functions detected in evaluation}$$

B ≡***Number of functions described in requirement specifications***

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Do functional tests (black box test) of the system according to the requirement specifications. Count the number of missing functions detected in the evaluation and compare with the number of the function described in the requirement specifications.	0<=X<=1 The closer to 1.0 is the better.	Req. spec. Evaluation report	Developer, SQA

4.1.2 Correctness

This attribute will be checked by the **<u>Interface standard compliance (ISC)</u>** Metric which measures compliance of interfaces to applicable regulations, standards ,and conventions. The metrics will be computed with the following equation.

$$\boldsymbol{ISC}\,\% = \boldsymbol{A} \div \boldsymbol{B}$$

$\therefore$ ***A= Number of correctly implemented interfaces as specified***

B ≡Total number of interfaces requiring compliance

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Count the number of interfaces that meet required compliance and compare with the number of interfaces requiring compliance as in the specifications.	0<=x<= 1 the closer to 1.0 is the better.	Product description of compliance and related standards, convention s or regulations test Specification and report	Developer

.

4.2 Reliability

Fault Tolerancewill be checked by the **<u>Mean Time Between Failures (MTBF)</u>**

Metric this measures the frequency of the software to fail in operation. The metrics will be computed with the following equation.

$$\mathbf{MTBFF\%} = X - Y$$

$$\therefore \boldsymbol{X = T1/A}$$

$$\boldsymbol{Y = T2/A}$$

A ≡ total number of actually detected failures (Failures occurred during observed operationtime)

$T1$ ≡operation time

$T2$ ≡sum of time intervals between consecutive failure occurrences

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Count the number of failures occurred during a defined period of operation and computes the average interval between the failures.	0<X, Y The longer is the better. As long time can be expected between failures.	Test report Operation (test) report	User

4.3 Usability

Understandabilitywill be checked **by <u>Understandable Input And Output (Un. I/O)</u>** Metric which measures the ability of new team developer to understand what is required as input data and what is provided as output by the software system. The metrics will be computed with the following equation.

$$\mathbf{Un.I/O} = \boldsymbol{A \div B}$$

$\therefore$ ***A ≡ Number of input and output data items which user successfully understands***

B ≡Number of input and output data items available from the interface

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Conduct user test and interview user with questionnaires or observe user behavior. And Count the number of input and output data items understood by the user and compare with the total number of them available for the user.	0<=X<= 1 The closer to 1.0 is the better.	User manual Operation (test) report	User

4.4 Efficiency

4.4.1 Time Behavior

This attribute will be checked by **<u>Throughput (Thr)</u>** Metric which measures the number of tasks can be successfully performed over a given period of time. This can be computed with the following equation.

$$\mathbf{Thr} = A/T$$

$$\therefore A \equiv \textit{number of completed tasks}$$

$$T \equiv \textit{observation time period}$$

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Calibrate each task according to the intended priority given. Start several job tasks. Measure the time it takes for the measured task to complete its operation. Keep a record of each attempt.	0 < X The larger is the better.	Testing report Operation report showing elapse time	Developer, SQA

4.4.2 Resource Behavior

This attribute will be checked by **<u>Mean time to response (MTR)</u>** Metric which measures the average wait time for the user experiences after issuing a request until the request is completed within a specified system load in terms of concurrent tasks and system utilization. This can be computed with the following equation.

$$\boldsymbol{MTR = Tmean \,/\, TXmean}$$
$$\therefore \boldsymbol{Tmean = \sum(Ti) \,/\, N, \qquad (for\ \ i = 1\ to\ N)}$$
$$\boldsymbol{TXmean = required\ mean\ response\ time}$$
$$\boldsymbol{Ti = response\ time\ for\ i - th\ \ evaluation\ (shot)}$$
$$\boldsymbol{N = number\ of\ evaluations\ (sampled\ shots)}$$

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Execute a number of scenarios of concurrent tasks. Measure the time it takes to complete the selected operation(s). Keep a record of each attempt and compute the mean time for each scenario.	0 <= X The nearer to 1.0 and less than 1.0 is the better.	Testing report. Operation report showing elapse time	User Developer

4.5 Maintainability

4.5.1 modularity

This attribute will be checked by **<u>Classes Dependency (Cl.Dep)</u>** Metric which measures the summation of dependency between system classes.This can be computed with the following equation.

$$\textbf{Cl. Dep\%} = \sum_{i=1}^{n} \boldsymbol{D.C/n}$$

$$\boldsymbol{D.C \equiv Dependency\ edges\ in\ ER\ \ diagram\ of\ \ the\ sharable\ site\ of\ Database}$$

$$\boldsymbol{n \equiv total\ number\ of\ edges\ in\ diagram}$$

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Review the ER diagram and customize it to semi diagram for shareable artifacts.	0 < X The larger is the better.	ER diagram, SRS	SQA

4.5.2 Analyzability

This attribute will be checked by the **strength of Document (Str.Doc)** Metric which measures the understandability and clarity of the documentation .This can be computed with the following equation.

$$\textbf{Str. Doc \%} = \boldsymbol{A \div B \times 100}$$

$$\boldsymbol{A \equiv number\ of\ functions\ that\ defined\ clearly\ in\ documentation}$$

$$\boldsymbol{B \equiv number\ of\ function\ that\ defined\ in\ SRS}$$

Method Of Application Measurement	**Interpretation Of Measured Value**	**Sources Of Input To Measurement**	**Target Audience**
Making a deep Review (inspection) for the SRS document and try to discover if there is an ambiguous function.	100 >= X The larger is the better.	SRS, Manual Help Document, online Help	SQA Team,Tester.

4.6 Variability

4.6.1 Component Variability

This attribute will be checked by the **Total Variability (TV)** Metric which measures the percent of the component variability .This can be computed with the following equation.

$$\boldsymbol{TV} = \frac{\sum(\boldsymbol{Fun}, \boldsymbol{Main}, \boldsymbol{Effiec}, \boldsymbol{reli}, \boldsymbol{usa})}{\text{each attribute devided by its weight}}$$

Attributes	**Weight**
Functionality	30%
Maintainability	10%
Usability	10%
Reliability	30%
Efficiency	20%

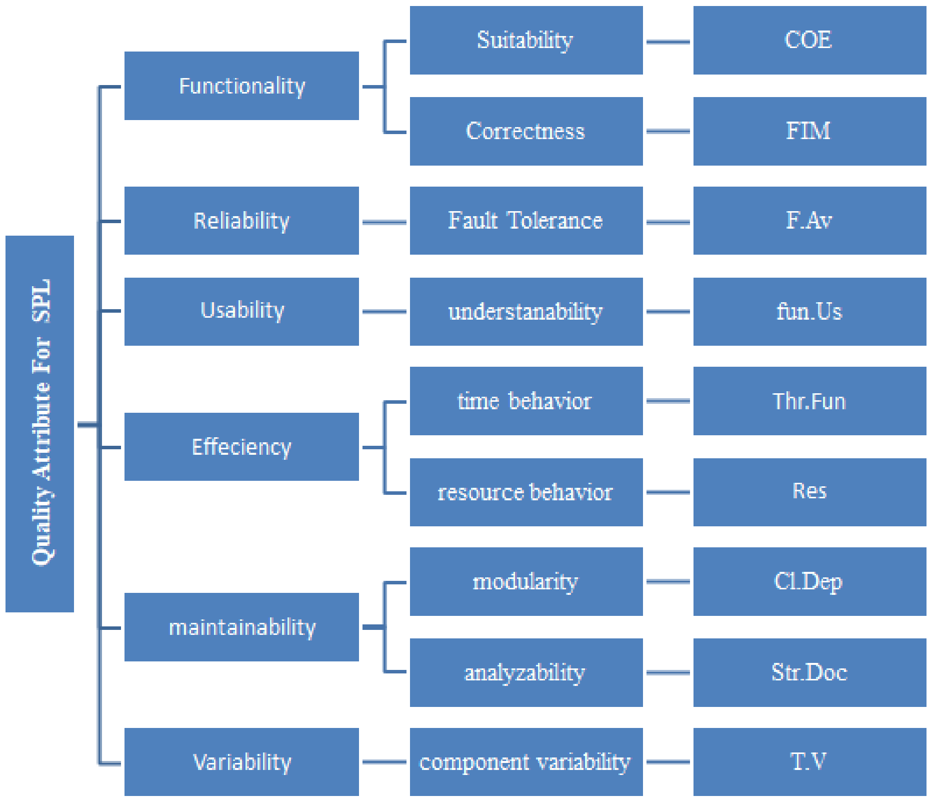


Figure (4-1) The Proposed Model

CHAPTER FIVE EVALUATION PROPOSED MODEL

CHAPTER FIVE
EVALUATION PROPOSED MODEL

This chapterpreviewthe application of the equations that were previously proposed for evaluation by applying the cognitive walkthroughmethod to the case study.

5.1 Evaluation

5.1.1 Functionality

5.1.1.1 Suitability

All the functions that are described in the specification were executed in the core asset that means. **FIC = 1-(0/13)=1-0=1** (which considered the better value).

.

5.1.1.2 Correctness

Mostof the specified functions implemented correctly excepttwofunctionsgetting un expectedresults and that make the **ISC**=11/13=0.84.

5.3.2 Reliability

5.3.2.1 Fault Tolerance

All system failure according to the database the **MTBF** = 33 d / 2 = 16.5 day/failure. But if we do not take the database failure in our view the **MTTR** = 0 h/break.

5.3.3 Usability

5.3.3.1 Understandability

All the inputs in the interfaces are described with agood naming convention and the **Un. I/O**=300/300=1(which considered the better value).

5.3.4 Efficiency

5.3.4.1 Time behavior

According to the table(5-1), we perform 7 tasks in 216 seconds and found that the **Thr** =A/T =7/216=0.032 tasks per second (which is considered a good performance).

Table (5-1) illustrates the result experiments for tester

#	**Begin time**	**First response**	**End of task**	**Difference between begin and end**	**Difference between begin and first response**
1	0	3	32	32	3
2	32	36	48	16	4
3	56	60	79	23	4
4	83	84	104	21	1
5	110	113	180	70	3
6	187	192	203	16	5
7	213	220	251	38	7
				Total=216	Total=27

5.3.4.2 Resource Behavior

Because we suppose that there is no exhaustive system we but one second as an assumption for response time for each function. According to the table(5-1), we found that the **MTR**= 27/7=3.857/7=0.55 second per function.

5.3.5 Maintainability

5.3.5.1 Modularity

After reviewing the ER diagram which shown in figure (7) we found that the total number of the table is 13 tables and the **Cl.Dep** = (11/13)*100=84%

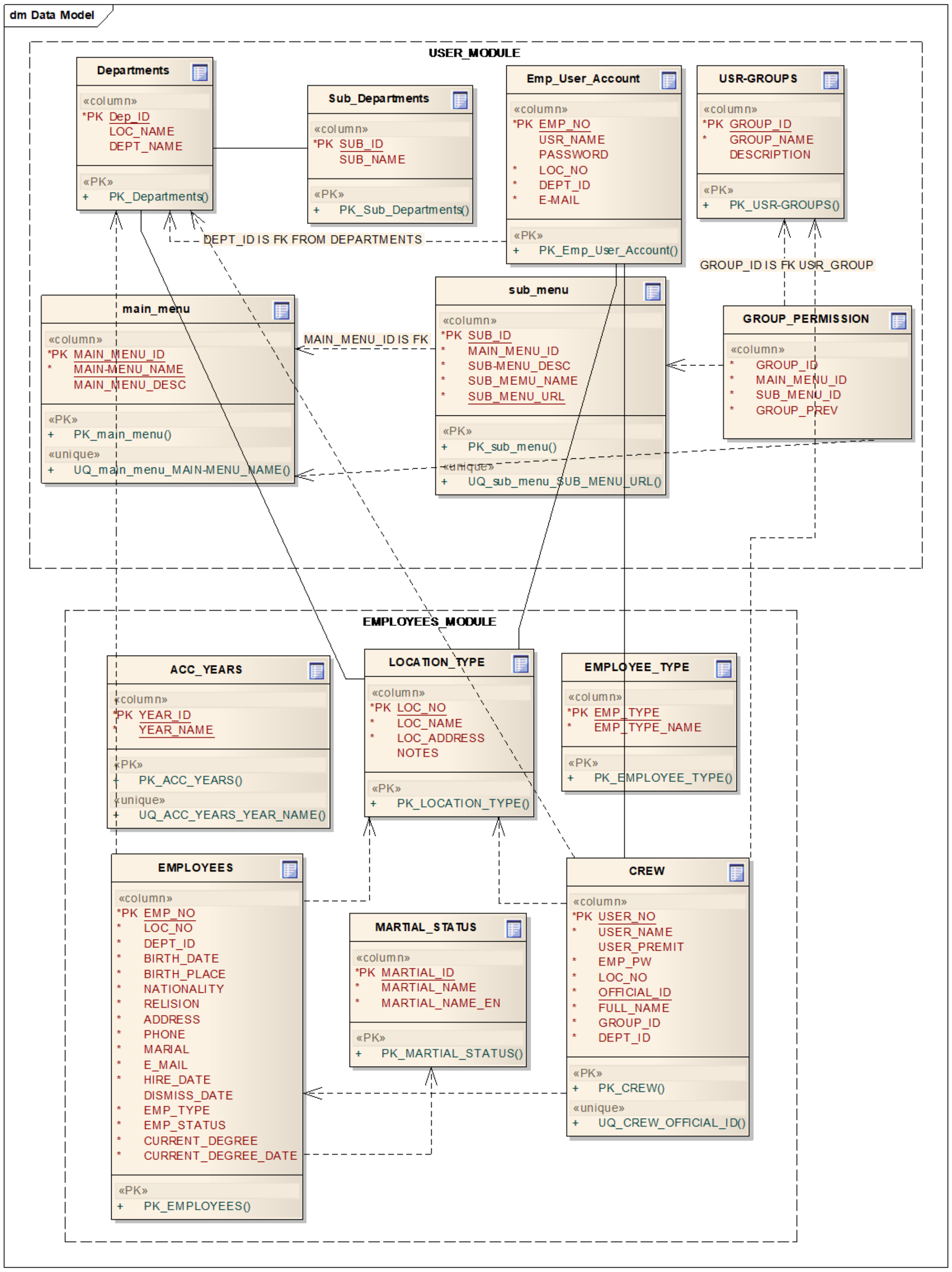


Figure (5-1) the ER diagram of the core assets in the system.

5.3.5.2 Analyzability

All the functions are declared clearly with Volare template so the **STr.Doc** =13/13=1*100=100 % (which is the best value as we defined before).

5.3.6 Variability

5.3.6.1 Component Variability

As shown in Table(5-2) the Total Variability (TV) founded equal 77.89 (which is considered a good value).

Table(5-2) Illustrate the variability of the component

Attributes	**Weight**	**Sub attribute**	**value**	**Value with weight**	**Summation of Value with weight**
Functionality	30%	Suitability	1	150	27.6
		Correctness	0.84	126	
Maintainability	10%	Modularity	84	42	9.2
		Analyzability	100	50	
Usability	10%	Understandability	1	10	10
Reliability	30%	Fault Tolerance	16 days(53.3)	159.9	15.99
Efficiency	20%	Time Behavior	0.032	96.8	15.1
		Resource Behavior	0.55	55	
Total					77.89

CHAPTER SIX
CONCLUSIONS AND FUTURE WORK

CHAPTER SIX

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The result of this research is a quality model dedicated to evaluate the variability of the abstraction level of Product Line Architecture(PLA) and the core assets of software product line systems.
The proposed model includes a set of quality attributes selected based on the benefit of SPL and the ISO/IEC 9126 model.
The metrics used to measure these attributes were selected and applied with the proposed model to the selected system (case study).
The result of the application approve the applicability of the model to get the variability as a main goal and to evaluate functionality ,uasbility ,reliability ,understandability as second goals . the proposed model can be used not just for evaluating component based systems artifacts .

6.2 Future Work

As a complement to this research, there are some recommendations for researchers in this subject by improving the quality model:

- Apply this model for component based systems for evaluation customizability of components .
- Add the commonality and traceability attribute to the quality model, because they are very important concepts in SPL.
- Extend the metrics that used to measure the attributes of the model.
- Apply the model in different similar cases or systems (using the same techniques) and comparing the results.
- Define a new model for the second phase of SPL (Application Engineering phases) .

References

Adam Trendowicz&Teade Punter, 2018.Quality Modelling for Software Product Lines .*FraunhoferIESE*.Sauerwein 6, D-67661 Kaiserslautern, Germany.

Adewumi, A., Misra, S. &Omoregbe, N., 2013.A Review of Models for Evaluating Quality in Open Source Software.*IERIProcedia*, 4, pp.88–92.

Al-Rawashdeh, T.A., Al'Azzeh, F.M. & Al-Qatawneh, S.M., 2014.Evaluation of ERP Systems Quality Model Using Analytic Hierarchy Process (AHP) Technique.*Journal of Software Engineering and Applications*, 07(04), pp.225–232.

Alvaro, A., Almeida, E.S.D. &Meira, S.L., 2007.Towards a Software Component Certification Framework.*Seventh International Conference on Quality Software (QSIC 2007).*

Boehm, B. W., Brown, H., Lipow,M ,1978 .Quantitative Evaluation of Software Quality .TRW Systems and Energy Group.

Clements, P. & Northrop, L., 2012.*Software product lines: practices and patterns* 3rd ed., Boston: Addison-Wesley.

Etxeberria, L., Sagardui, G. &Belategi, L., 2008. Quality aware software product line engineering. *Journal of the Brazilian Computer Society*, 14(1), pp.57–69.

Farooq , S.U., K, S.M. &Nesar Ahmad, 2014. SOFTWARE MEASUREMENTS AND METRICS: ROLE IN EFFECTIVE SOFTWARE TESTING .*International Journal of Engineering Science and Technology* , 3.

F. Bachman, M. Goedicke, J. Leite, R. Nord, K. Pohl, B. Ramesh, A. Vilbig , 2003 . Managing Variability in Product Family Development, the 5th Workshop on Product Family Engineering (PFE-5), published in Springer Verlag Lecture Notes on Computer Science .

Glott, R. et al., 2010. Quality Models for Free/Libre Open Source Software Towards the "Silver Bullet"? *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications* .

GEORGIADOU, E.L.L.I., 2003. GEQUAMO—A Generic, Multilayered, Customisable, Software Quality Model.*Software Quality Journal*.

Heradio, R. et al., 2016. A bibliometric analysis of 20 years of research on software product lines.*Information and Software Technology*, 72, pp.1–15.

John McGregor, 2001. Testing a Software Product Line.*Software Engineering Institute* .

Mc Call, J. A. &, Richards, P. K. & Walters, G. F,1977.Factors in Software Quality Volumes I, II, and III.US Rome Air Development Center Reports, US Department of Commerce, USA.

M. Clauss , 2001 . Modeling variability with UML, GCSE 2001 - Young Researchers Workshop,September 2001.

Miguel, J.P., Mauricio, D. & Rodríguez, G., 2014.A Review of Software Quality Models for the Evaluation of Software Products.*International Journal of Software Engineering & Applications*, 5(6), pp.31–53.

Montagud, S., Abrahão, S. &Insfran, E., 2011. A systematic review of quality attributes and measures for software product lines. *Software Quality Journal*, 20(3-4), pp.425–486.

M. Jazayeri, A. Ran, F. van der Linden , 2000 . Software Architecture for Product Families: Principles and Practice, Addison-Wesley.

Rawashdeh, A. &Matalkah, B., 2006.A New Software Quality Model for Evaluating COTS Components.*Journal of Computer Science*, 2(4), pp.373–381.

R. van Ommering , 2000 . A Composable Software Architecture for Consumer Electronics Products, XOOTIC Magazine.

SEI, 2016. Key Benefits: Why Product Line Engineering? *Product Line Engineering: Key Benefits*. Available at: http://www.productlineengineering.com/benefits/key-benefits.html [Accessed November 19, 2018].

Samoladas, I. et al., 2008. The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation. *IFIP – The International Federation for Information Processing Open Source Development, Communities and Quality*, pp.237–248.

S. Thiel, A. Hein , 2002. Systematic integration of Variability into Product Line Architecture Design, second International Conference on Software Product Lines(SPLC-2).

Umm-E-Laila et al., 2017.Comparison of open source maturity models.*Procedia Computer Science*, 111, pp.348–354.