Sudan University of Science and Technology

College of Post Graduates Studies

A Framework for Evaluating Reusability of Core Assets using SPL and SOA

إطار لتقويم إعادة الاستخدام للأصول الأساسية باستخدام خط إنتاج البرمجيات والمعمارية خدمية التوجه

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

November 2018

Sudan University of Science and Technology

College of Post Graduates Studies

A Framework for Evaluating Reusability of Core Assets using SPL and SOA

إطار لتقويم إعادة الاستخدام للأصول الأساسية باستخدام خط إنتاج البرمجيات والمعمارية خدمية التوجه

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

Prepared By: Alaa Ramadan Suliman Ali

SUPERVISORS:

Dr. Almahdi Ibrahim
Dr. Arafat Abdulgader
Date……./...…. /…...

الآية

﴿ ألم تَرَ أنَّ الله أنزلَ من السَّماء ماءً فأخرجنا به ثمراتٍ مُخْتلفاً ألوانُها ومن الجبالِ جُدَدٌ بيضٌ وحمرٌ مختلفٌ ألوانُها وغرابيبُ سُودٌ ﴾

صدق الله العظيم

سورة فاطر: الآية 27

DEDICATION

Special dedication to my family members especially to my beloved father and mother who always give me the encouragement in my life, my study and the support to finish my project.

To my Supervisor

Dr. ALmahadi Ibrahim

To all SUST’s teachers

and all my friends out there

Thank You for your supporting and teaching

Thank You for everything you gave during my studies and the knowledge that we

shared together.

THANK YOU SO MUCH

ABSTRACT

Software reusability is one of the quality attributes that illustrate the importance of software to software developers and the return of investment for this software. Accordingly, it is very important taking into consideration all sub attributes which may affect the calculation of final reusability value, therefore it is good to provide a framework for evaluating reusability based on sub attributes that have direct contribution in reusability value. The research focus on designing framework for evaluating reusability in software product line (SPL) and service oriented architecture (SOA) approaches, where the two approaches are supporting the reusability concept. This is expected to be greatly reused when the two concepts are combined. All that mentioned above because the current quality models do not address the reusability of most important characteristic resulting from the integration of the two concepts, which are the core assets from software product line methodology and the web service from service oriented architecture methodology.

The framework was designed by selected quality attributes from key feature of core assets and web service, also defined metrics for each attributes, and then applied the framework on the selected target system. Finally calculated the final result of reusability after applied the framework steps. The designed framework defined a systematic method for calculating the reusability of core assets as web service, where each attribute is calculated by reference to the related metric and artifact, which artifacts were defined previously before applying the framework. As in research case study, the case study was defined according to FODA, FAST and KOBRA methodology, and according to WSDL, XML and SOAP standard. Thus, all attributes was contributing in calculating the final value of reusability. At last, studied the final framework result to make an appropriate decision.

The framework was applied to the Bank third party services which are credit recharge, electricity buying and water billing. The final result was obtained is 82.8% and is considered a high value compared with reusability values used in previous studies when applying the reuse calculation model for each methodology.

المستخلص

خاصية إعادة إستخدام البرمجيات واحدة من خصائص الجودة التي توضح أهمية البرمجيات لمطوري البرمجيات والعائد من هذه البرمجيات. وفقا لذلك نجد أنه من المهم جدا ان نضع في الاعتبار كل الخصائص الفرعية التي تؤثر أو تساهم في القيمة النهاية لإعادة الإستخدام, لذا من الجيد وجود إطار يقوم بتقييم إعادة الإستخدام بالإعتماد علي الخصائص الفرعية التي لها تأثير مباشر علي قيمة إعادة الإستخدام النهائية. البحث يركزعلي تصميم إطار لتقييم إعادة الإستخدام في منهية خط إنتاج البرمجيات والمعمارية خدمية التوجه, المنهجيتان تدعمان مفهوم إعادة الإستخدام. ويتوقع إعادة إستخدام بشكل كبير عند دمج المفهومين معا. كل ما ذكر أعلاه لأن نماذج الجودة الحالية لا تدرس إعادة الاستخدام لأهم الخصائص الناتجة من دمج المفهومين, والتي تتمثل في الأصول الأساسية التابعة لمنهجية خط إنتاج البرمجيات وخدمة الويب التابعة لمعمارية خدمية التوجه.

تم تصميم الإطار بإختيار خصائص الجودة من المعالم الأساسية للأصول وخدمة الويب, وأيضا تم تعريف معادلات لكل خاصية, وبعد ذلك طبق الإطار علي النظام الذي تم إختياره كدراسة حالة. وأخيرا تم حساب القيمة النهائية لإعادة الإستخدام بعد تطبيق خطوات الإطار. الإطار المصمم يوضح طريقة منظمة لحساب إعادة إستخدام الأصول الأساسية كخدمة ويب, حيث يتم حساب كل خاصية بالرجوع إلى المعادلة المقابلة لها وأيضا بالرجوع إلى الوثيقة أو المستند, أي مستند يتم تعريفه سابقا قبل تطبيق الإطار. كما في حالة دراسة الجدوي, تم تعريف المستندات بإتباع منهجية ال KOBRA وال FODA وال FAST وأيضا بإتباع معيار ال WSDL وال XML وال SOAP. وهكذا نجد ان كل الخصائص ساهمت في حساب القيمة النهائية لإعادة الإستخدام. وأخيرا تدرس نتيجة الإطار النهائية لإتخاذ قرار مناسب.

تم تطبيق الإطار علي خدمات البنك الثانوية (خدمة شراء الكهرباء وخدمة شحن الرصيد وخدمة دفع فواتير المياه), وكانت نتيجة إعادة إستخدامها 82.8% وتعتبر قيمة عالية مقارنة بقيم إعادة الإستخدام التي تناولتها الدراسات السابقة عند تطبيق نموذج حساب إعادة الإستخدام لكل منهجية على حدى.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Terms
SPL	Software Product Line
SOA	Service Oriented Architecture
PLE	Product Line Engineering
ROI	Return Of Investment
PL	Product Line
SPLE	Software Product Line Engineering
DRM	Decision Resolution Model
XML	Extensible Markup Language
WSDL	Web Services Description Language
BPMN, BPEL, UDDI, WS-Security, OAI-PMH, SRW/U and ESB	Web Service Standards
SynCSI	Syntactic Completeness of Service Specification
SemCSI	Semantic Completeness of Service Specification
FC	Function Coverage
NFC	Non-Functional Commonality
AC	Architectural Commonality
NC	Nonfunctional Coverage
MD	Modularity
CV	Coverage Variability
CA	Cumulative Applicability
SC	Standard Conformability
TL	Tailorability
VP	Variation Point
ET	Effectiveness Tailoring
TC	Tailorability of Closed variability
TO	Tailorability of Open variability
CC	Component Compliance
DC	Discoverability
RE	Reusability
KOBRA, FODA and FAST	Software Product Line Methodologies

Chapter One

Introduction

1.1 Introduction

Software Product Line (SPL) and Service Oriented Architecture (SOA) are one of the recent and effective reuse approaches, where applications are generated by instantiating a core assets which are a large-grained reuse unit in SPL, also applications are built by subscribing web services which are most popular SOA implementation that can be reused by web service consumers. Hence, core assets and web services are a key element of SPL and SOA sequentially, therefore reusability of the core assets based web service largely determines the success of projects [13].

However, the current quality models aren't adequately addressing the reusability for unique characteristics of incorporate two concepts core assets and web services.

Our argument to demonstrate this research that how this framework helps in achieving the better reusability and high return on investment.

1.2 Problem statement

The main research problem that the current quality models do not address the reusability of most important characteristic resulting from the integration of the two concepts. AS result for that :

- Industrial practice is still in its infancy and applying the previous model hasn't had a direct impact on investment.
- The evaluating criteria were built for reusing core asset of a single programming language.

1.3 Significance

This research provides these main advantages:

- Assess reusability of software systems before publishing in market.
- Ensure reusing core assets regardless of the programming language.
- Define weakness point that helping enhances the effectiveness of reuse.
- Giving clear decision about the feature of product.

1.4 Hypothesis

The derived attributes from key feature of core assets and web services, which support maximum reuse resulting in calculation high reusability value.

1.5 Objectives

1- Investigate in evaluation reusability of core asset web service in software product line based service oriented architecture approach.
2- Select quality attributes which supports reusability purpose.
3- Define metric for each attributes.
4- Develop framework to evaluate reusability of core asset web service using software product line and service oriented architecture approach.
5- Evaluate our framework in light of research domain.

1.6 Scope

This research for building reusability framework based on SPL and SOA methodologies, the framework is applied on core asset web service. Framework attributes have relation with reusability of core asset and web service. We don't include the treatment of return of investment ROI.

1.7 Thesis layout

The research contains five chapters as shown below:

Chapter one: Introduction

It gives introduction about the project, defining the problem, significance, hypothesis, objectives and Scope of research.

Chapter two: Literature reviews and related work

It consists of two parts; part one represents a general background about the topic and parts two is the related studies.

Chapter three: Methodology

It describes the mechanisms used in this research from selecting key feature characteristic to define metrics for derived attributes, then building the reusability framework and defining the case study.

Chapter four: Framework implementation and result

It is representing the applying framework in case study, show the result of reusability and detecting from hypothesis.

Chapter five: Conclusions and Recommendations

Chapter Two

Literature Review & Related Work

2.1 Introduction

This chapter consists of two parts; part one represents a general background in software product line, service oriented architecture, combine software product line with service oriented architecture approaches and software. Part two shows the related studies.

2.2 Software Product Line

A software product line (SPL) is defined as a set of software systems sharing explicitly defined and managed common and variable features and relying on the same domain architecture to meet the common and variable needs of specific markets [4]. It is also defined as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of particular market segment or mission and that are developed from a common set of core assets in a prescribed way [5]. There is different process models exist for the development process of product lines mention in [6], the common one is software product line engineering (SPLE) is a systematic method to develop and use a software product line for a particular product domain, which consists of two essential activities, the first is domain engineering (product line architecture) to develop the core assets, the second is application engineering (product line members) to develop products using the core assets. SPLE provides guidance for the three essential activities, including 29 good practices, which are categorized into three practices areas: software management, technical management, and organizational management [7].

Product line scope and product line analysis define the boundaries and requirements of the software product line, based on the business goals of the organization. The scope identifies the characteristics of the defined systems as being inside or outside the boundaries of the software product line. The scope also helps identify the common aspects of the systems, as well as expected ways in which they may vary [13].

Composition elements of product line products are the product line core assets artifacts or resources used in the production of products in an SPL. A core asset may be the architecture, a software component, a process model, a plan, a document, or any other artifact useful in building a system. Each core asset has an attached process that describes how the core asset is used in product production, including variant information. For example, the product line architecture must include mechanisms to support the explicitly allowed variations in the products within the product line scope; the attached process defines how that variation mechanism is used to generate the architecture for a specific product variant [13].

2.2.1 Meta-model of core asset

The term, core asset, has been defined largely conceptually reuse asset that captures commonality and variability among products in product line. As shown in figure (2.1), it is plays a key role in PLE, and it has architecture which is generic to products, a component model capturing components and interfaces, and a decision model defining variability realization. We assume that it is detailed design model not implement source code [6].

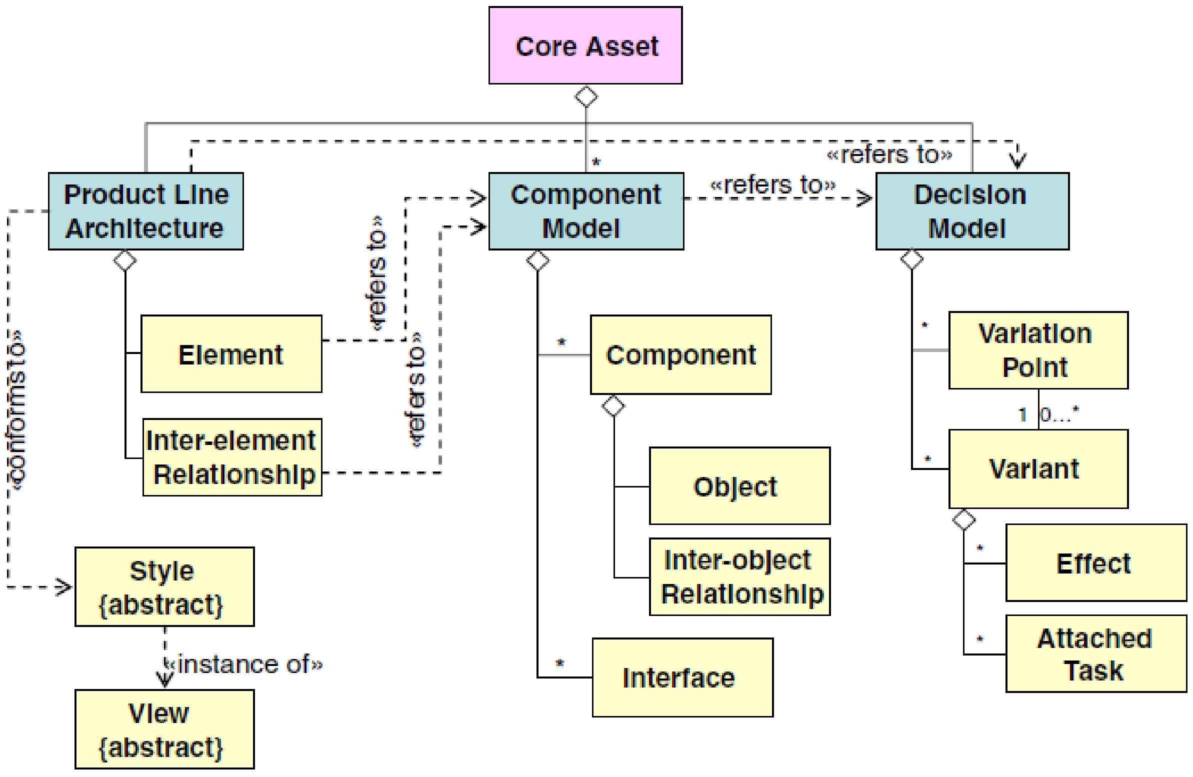


Figure (2.1) Meta-model of core asset [6].

Product line (PL) architecture represents architectural decisions that are common among applications in a product line. It is models and realizes architectural decisions that are variable among applications. Hence, it is used as a reference architecture that would be instantiated for the specific requirement of each application. According to the perspective to highlight the PL architecture, views such as Module View, C&C View and Deployment View can be decided. Several styles of a view can be applied to model PL architecture. Two main elements of PL architecture, element and inter-element relationship, are derived from the requirements and guided by styles. These elements further refer to components and relationships of component model [6].

Component model realizes functionality derived from functional and non-functional requirements of a core asset. Therefore, the component model is referred from elements and inter-element relationships in PL architecture. A component is represented with structural and behavioral model of objects, inter-object relationships, and interfaces it conforms to. Components also realize variable requirements among applications in a product line [6].

A decision model is a specification of variability in a core asset, and it consists of variation points, their associated variants, effects, and attached tasks. A variation point is a place where there exists a minor variation and variants are valid values which fill in a variation point. Effects are results of setting the variants, which are post conditions, side effects, or affected variation points. Attached tasks are action tasks or resolution mechanisms that have to be performed to set the variant. Variability listed in the decision model is eventually reflected and realized in a PL architecture, component models and interfaces. PLE supports closed and open variability since potential applications as well as known applications are considered. To generate applications based on a core asset, it is instantiated by resolving variability of a core asset by applying a Decision Resolution Model (DRM) which specifies application specific variants. A PL architecture and a component model of a core asset are specialized with a DRM [6].

2.2.2 Software product line approach

Adopting SPL approach implies performing two main activities [25]:

- Domain Engineering

  This activity is twofold:

  - Collecting, organizing, and storing past experiences in building systems in the form of reusable assets in a particular domain.
  - Providing an adequate means for reusing these core assets when building new systems.

  The term developing for reuse is often used to characterize the Domain Engineering. It can be divided in three main processes: Domain Analysis, Domain Design, and Domain Implementation. The domain analysis consists in capturing information and organizing it as a model. The domain design consists in establishing the product line architecture. The domain implementation consists in implementing the architecture defined during the domain design as software components.
- Application Engineering

  This activity consists of building systems based on the results of Domain Engineering. The requirements are selected from the existing domain model, which matches the customer’s needs. We assemble applications from the existing reusable components. The application engineering activity consists in building systems based on the results of Domain Engineering. During application requirements of a new system, we select the requirements from the existing domain model, which matches the customer’s needs. We assemble applications from the existing reusable components. The term developing by reuse is used to characterize the application engineering activity.

2.3 Service Oriented Architectures

"SOA is a conceptual business architecture where business functionality, or application logic, is made available to SOA users, or consumers, as shared, reusable services on an IT network. 'Services' in an SOA are modules of business or application functionality with exposed interfaces, and are invoked by messages" [8]. A SOA is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Component-based development proceeds by composing software systems from pre-fabricated components (often third-party black-box software). A typical component-based system architecture comprises a set of components that have been purposefully designed and structured to ensure that they fit together (i.e. have pluggable interfaces) and have an acceptable match with a defined system context. Service-oriented development on the other hand proceeds by integrating disparate heterogeneous software services from a range of providers [9]. A SOA is a means of designing software systems to provide services to either end user applications or other services through published and discoverable interfaces.

2.3.1 Key Elements of a Service-Oriented Architecture

A Service-oriented Architecture comprises several key elements. Its elements work together to more closely link business needs with IT. The following list covers the essential ingredients of an SOA [8]:

- **Conceptual SOA vision** – An SOA is a business concept, which includes clearly defined business, IT and architectural goals.
- **Services** – An SOA enfolds all possible services in the organization alongside a service design model to assure reusability, interoperability and integration across all business processes and technology platforms. Services are indeed the central artifact of a Service-oriented Architecture.
- **Enabling technology** – The technology must ensure that your services operate reliably and securely in support of the stated business objectives.
- **SOA governance and technologies** – The SOA governance model defines the various governance processes, organizational roles, standards and policies adhered to in the conceptual architecture.
- **SOA metrics** – The SOA metrics include Service-Level-Agreements (SLAs) for individual services, as well as usage metrics, business and return on investment metrics as well as process metrics.
- **Organizational and behavioral model**

Generally, systems based on an SOA have many users and providers, where certain users also act as providers to other users. Most SOA- systems are considerably more complex than the one in Figure (2.2), which illustrates the most basic SOA architecture, but they all follow the same basic principles.

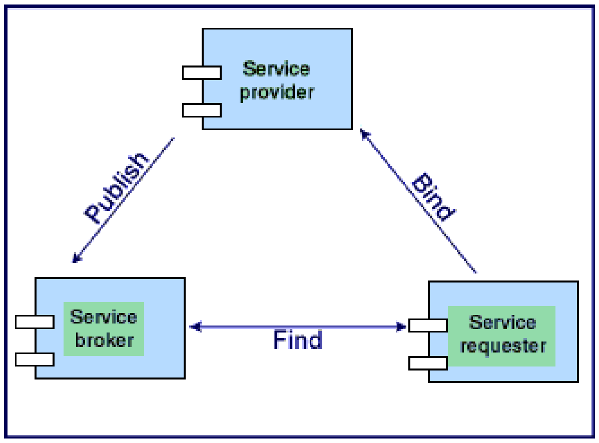


Figure (2.2) Basic SOA architecture [8].

2.3.2 Basic and Architectural Principles of a Service Oriented Architecture

There are several guiding principles that define the ground rules for development, maintenance and usage of the SOA. The guiding principles cover [10]:

- Reuse, granularity, modularity, composability, componentization, and interoperability.
- Compliance to standards (both common and industry-specific).
- Services identification and categorization, provisioning and delivery, and monitoring and tracking.

The following specific architectural principles for design and service definition focus on specific themes that influence the intrinsic behavior of a system and the style of its design. They are derived from the guiding principles and cover [11]:

- **Service Encapsulatio**n - Accessing functionality through some well-defined interface, the application being seen as a black box to the user.
- **Service Loose coupling -** Services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other.
- **Service contract -** Services adhere to a communications agreement, as defined collectively by one or more service description documents.
- **Service abstraction -** Beyond what is described in the service contract, services hide logic from the outside world.
- **Service reusability -** Logic is divided into services with the intention of promoting reuse.
- **Service composability -** Collections of services can be coordinated and assembled to form composite services.
- **Service autonomy** – Services have control over the logic they encapsulate.
- **Service statelessness** – Services minimize retaining information specific to an activity [10].

 Service discoverability – Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

So we can say the SOA infrastructure is connection mechanism between services and service consumers. It usually implements a loosely coupled, synchronous or asynchronous, message-based communication model. The infrastructure often contains elements to support service discovery, security, data transformation, and other operations. One common SOA infrastructure is an enterprise service bus (ESB) to support SOA environments [13].

2.4 Combining SPL and SOA Approaches

Meeting business goals through a product line or a set of service-oriented systems requires variation management of assets, including services. Variation management — comprises all activities to explicitly model, manage, and document those parts, which vary among the products of a product line. Variation management applies to services in both the service-oriented and product line contexts. In both types of systems, variation points may be implemented either in a single service (where a service interface may offer parameterization or some other variation mechanism) or through similar services to address variation [13].

Whereas variation and variation management are key to an SPL approach, they are not highlights of an SOA approach. Yet, SOA approach can be employed as a variation mechanism when core assets and products are developed in a product line. In an SPL approach, developers of software core assets would package desired capability as a service. That service may have built-in variation points that are accessible through parameterized service calls, or the service registry may identify variations among related service components that may include both the newly packaged capabilities and existing services from the enterprise [13]. The SOA infrastructure provides an invocation mechanism to the needed service core asset for product development. Variation management in this context allows the tailored use of services to provide the exact, desired capability for a specific product. The dynamic nature of service invocation may support adaptation and more dynamic growth of a product line's scope. SOA may also support a more opportunistic response to changing market conditions with product line adaptation or new product lines. All mention above support and increase the concept of reusability [13].

2.5 Software Reusability

Software reusability refers to the probability of reuse of software. Another definition, "characteristics of an asset that make it easy to use in different contexts, software systems, or in building different assets”. The potential benefits of software reuse and the maturity of reusability concepts leads us to think about how we might measure them [10].

The definition of service reusability is derived from conventional definition on reusability which is 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 [12].

2.6 Related Work

2.6.1 A framework for evaluating reusability of core asset in product line engineering

Jin, Ji, Sang, Sung and Soo [2] presented comprehensive framework for evaluating the reusability of core assets. They drove a set of quality attributes that characterizes the reusability of core assets depending on key characteristics of core assets (functional commonality, applicability, non-functional commonality, variability richness, and tailorability) and ISO/IEC 9126 framework (Understandability and component replaceability). Then, they define metrics for each quality attribute and finally present practical guidelines for applying the evaluation framework in PLE. Moreover, they applied the metrics with case study (Rental core asset) which is developed using PLE process, DREAM. And get result that the rental core asset is 81.9% reusable. To validate the metrics, they performed theoretical analysis according to Kitchenham’s approach (Barbara Kitchenham, Shari Pfleeger, and Norman Fenton’s validation framework) and assessed the proposed framework in the perspective of six criteria which are derived by Ejiogu’s criteria and validity criteria of IEEE Std 1061[2].

The reusability rate of above quality framework may be increased If reusing core assets isn’t depend on specific environment and every family member able to operate the functions in his own environment.

2.6.2 A Quality Model for Evaluating Reusability of Services in SOA

Si Won Choi and Soo Dong Kim [12] presented comprehensive quality model for evaluating reusability of web services. They defined Key Features of web services in SOA and derived from it reusability attributes such as business commonality, modularity, adaptability, standard conformance and discoverability. In addition, they defined metrics for all attributes to collect the reusability. At last they applied this model on the domain of flight management which includes Flight Inquiry service, Flight Reservation service, and Flight Purchase service [12].

The final reusability value is 0.23, so it is very low compare with the expected output from SOA approaches, therefore there is very important individual attributes (business commonality) which it need to be revised more because it have direct effect in final reusability value.

2.6.3 Reusability Assessment of Open Source Components for Software Product Lines

Mahmood, Ahmad and Alan [16] presented exploratory study to explore the factors affecting the reusability of open source software (OSS) in SPL environment. They defined reusability assessment model contains six attributes related to the reusability of an SPL component: flexibility; maintainability; portability; scope coverage; understandability; variability. These emerged from the exploratory study (using interview and result was obtained using the grounded theory approach). These attributes are selected due to their internal nature. Also, they define metrics for any attribute, most of these came from literature review, however, a small number were devised by their selves like variability metric due to there is no measure available to variability. In addition to all that, they validate the metrics by conducting survey to collect data using questionnaire and compare results of reusability assessment by human evaluators against proposed model [16].

2.6.4 A survey and Proposed Reusability Assessment Framework for Aspect Oriented Product Line Core Assets

Mahmood, Ahmad [17] presented a survey of aspect oriented implementation of SPL reusability evaluation. They spoke about two frameworks one for reusability of core assets and the second for reusability and maintainability of aspect oriented program they address the lake in both and proposed new frame work but not implemented physically [17].

2.6.5 Combining Service Orientation with Product Line Engineering

Lee, Jaejoon, and Gerald [18] integrated the concept of software product line engineering(SPLE) with service oriented(SO) to create a new approach which is a service-oriented product line (SOPL) is a dynamic software product lines DSPL application domain that's built on services and a service-oriented architecture to develop core assets. The paper spoke about challenges to build SOPL approach which is:

- Different Notions of First-Class Objects (feature and service are two different notions of key engineering drivers in software development under the SPLE and SO paradigms). As result of comparison between feature and service that is dynamic and automated feature binding considering the features' quality attributes is basically missing in SPLE. And product line variations are difficult to capture explicitly using the notion of services in SO.

- Dynamic characteristics of a service-based system, the dynamic characteristic of SO is closely related to Quality of service(QoS) and dynamic-service orchestration, SPLE usually addresses quality issues statically during system design and implementation depend on Static quality management approaches(predicting resources of constituent view). So they define QoS in terms of features (define maximum limit of available resources) to select available services at runtime when starts negotiating with service providers. In addition to all that, Most SPLE focus on configuring product line variations before deployment and don't consider dynamic-service composition, so they specify static services, along with the tasks that constitute them, as workflows, and thus also specify these services' pre- and post-conditions, invariants, and dynamic-service interfaces. Finally, by integrating and parameterizing dynamic services at runtime, their solution lets users access static services with dynamic ones.
- Involvement of Third-Party Service Providers which was out of scope for SPLE but the closest thing to it might be the use of commercial off-the-shelf (COTS) components. Third-Party Service Providers have several advantages in (SO), including service negotiations, service monitoring, and service reputation systems which SOPL should incorporate them. Therefore, we propose a QoS-aware framework that provides automated runtime support for them.
- Variation Control and Management aim to provide flexibility through a layered structure and modular components, called bricks. They establish an explicit mapping relation between features and architectural components (bricks) so that selecting the features for a product generates a corresponding product configuration. After that the developer integrates reusable components into each brick by following the specifications described in the components.

2.6.6 Software product line case study for bank third party services Water billing, Electricity buying and Credit recharge

The research [19] spoke about developing SPL using kobra methodology and other methodologies like FODA and FAST to develop part of system, the use of kobra is to consolidate reusability and developing strong architecture system. Those methodologies are used to develop software product line that provide essential service for bank third party services, it applied on credit card recharge system, electricity buying services system and water billing system [19].

The product comply the recent era demand to family member with short delivery time but just in android and java language.

Chapter Three

Methodology

3.1 Introduction

This chapter is representing the methodology applied in this research to build reusability framework, which include defining the reusability framework attributes from key feature of web services in service oriented architecture and core assets in software product line, also including defining metrics for each attribute derived from characteristics and defining steps for applying the framework to calculate the final value of reusability. Finally represent introduction about case study plus artifacts used for measuring each attributes.

3.2 Methodology

Reusability framework for core assets web service is derived from two reusable model one belong to software product line approach and other belong to service oriented architecture approach. So below steps which it was followed to build the final framework, as shown in figure (3.1):

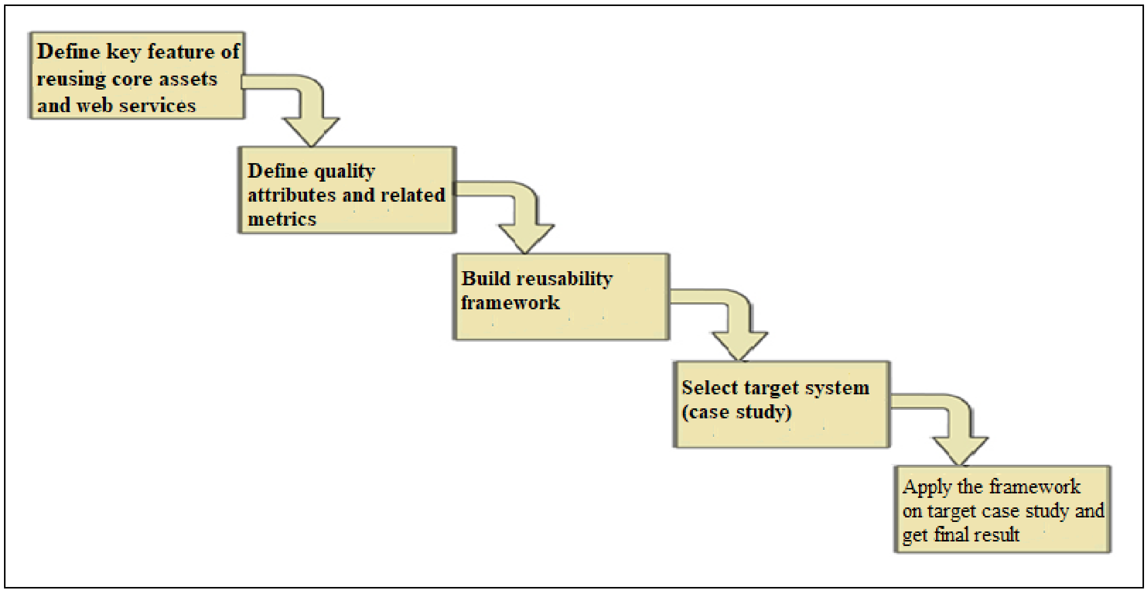

Define key feature of reusing core assets and web services
Define quality attributes and related metrics
Build reusability framework
Select target system (case study)
Apply the framework on target case study and get final result

Figure (3.1) explains the structure of working mechanism.

1. **Define key features of core assets and web services**, to define well applicable evaluation framework we must define all key features may effect on reusability of core assets in SPL and web services in SOA.
2. **Define quality attributes and related metrics**, after defining the key features for both core assets and web services we derived quality attributes from these features. Then define direct metrics for each attributes.

3. **Build reusability framework**, defining steps to apply the framework and all conditions before apply the frame work.
4. **Select the target system**, give introduction about bank third party services and how it works as web service. In addition give reason for selecting this system as case study.
5. **Apply the framework on target case study and get final result**, getting the last result of reusability by applying reusability framework steps at target case study.

We select the mention above two model duo to their popularity for reusing. And select the reusability attribute due to it's important after studying systematic review of quality attributes and measurement [14].

3.2.1 Key Features of core assets in software product line and web services in service oriented architecture

To define a reusability framework for evaluate core assets web service; key features should first be identified, from [12, 2] we derive the following features:

1- **Providing common functionality**: by developing and publishing common core asset which should capture common functionalities among family members to increase the inter-organizational consumer reuse. A set of family members consumer and common functionalities that will be provided by the core asset are defined in a PL scope [2].

2- **Capturing variability among products**: We should design the core asset to invoke each product of families by capturing variability. As shown in figure (3.2), the boundary of variability is not independent of commonality but within commonality. Variability is a minor variation within commonality and is composed of closed and open variability [2]. A mechanism to capture variability in a product line is a decision model. A decision model is a specification of variations in core assets and includes variation points, variants, effects, and attached task [2].

3- **Providing architectural genericity**: A core asset consists of components and their relationships that are affected by PL architecture. Software architecture is used for developing a single application. However, PL architecture of a core asset is used for developing a number of applications. Also, PL architecture includes architectural variability to be used by various applications. In addition, in PLE literatures, architecture in PLE is defined as a single element of a core asset. Such architecture will describe architectural aspects for a set of target applications in the product line. Here, the architecture is defined separately from internal design of components (called component model), and the decision model [2].

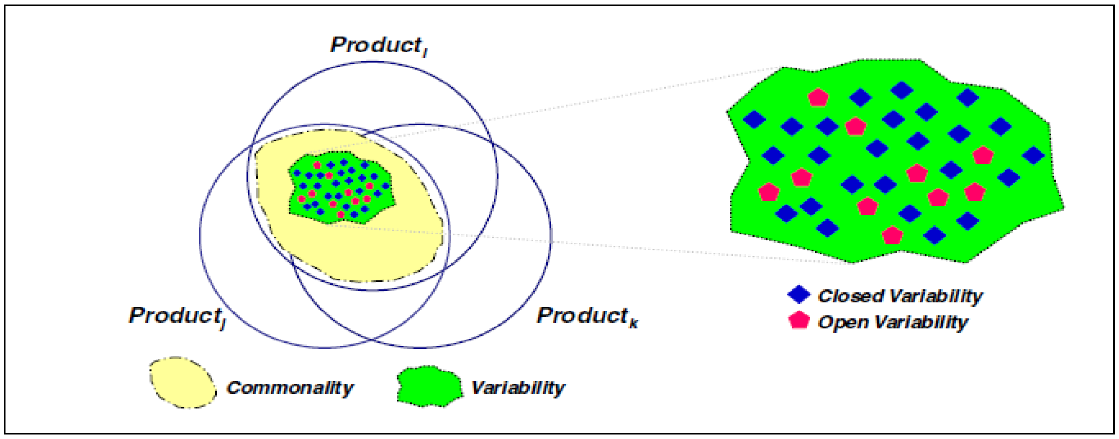


Figure (3.2) Common functionality and variability [2].

4- **Application-level core asset**: A core asset defined in this paper is a reuse unit larger than a component but smaller than a complete application and it is composed of PL architecture, components, and decision model. Also, a core asset is developed to be reused by several family members in the same domain. Thus, we can develop applications by instantiating a single core asset service and adding application specific information to the instantiated core asset service. Therefore, we call the granularity of a core asset service as the application level [2].

5- **Supporting open variability as well as closed:** Scope of variability in PLE is the cardinality of the variants for each variation point, which can be closed and open depending on the identified variants. For the closed scope, the variants for the variation point are known, and for the open scope, the variants for the variation point are unknown. Since the open scope does not decide variants, it is possible to add new variants for unknown application [2]. On the other side, closed variability fixes variants for applications.

6- **Instantiation mechanism for core asset to target applications:** In the view of a variability resolution mechanism, PLE defines its own term for the mechanism. Core asset service is instantiated to application specific assets by resolving variability and it is called instantiation [2]. The main activity of instantiation is to define DRM and bind the defined variants to each variation points.

7- **Embedding components:** To define a practical quality model, a core assets web service needs to be defined as a collection of concrete elements such as objects or components. In this research, the core assets web service case study includes a set of components and some of them can be replaced by in-house components or COTS components that are supplied by the third party. Since component follows the standard interface, we can replace designed component to other components when only functionality of designed component matches with functionality of the existing component and when component conforms to the component model [2].

8- **Well-defined Interfaces:** Core assets web service defines by selecting multiple features belong to one application, and they provide a well-defined interface which allows service consumers to use capabilities of services [12].
9- **Modularity of Services:** core assets web service is a highly modularized unit of capability, so they can be composed in various business processes without much complication [12].
10- **Loosely-Coupled Nature:** Providers publish core assets web service and consumers subscribe the core assets web service without an advanced knowledge on the other party [12].
11- **Standardization:** Due to the heterogeneity, conformance to SOA standards becomes essential. This is the reason why there exist a number of standards in SOA such as BPMN, BPEL, WSDL, UDDI, SOAP, and ESB [12].
12- **Subscription-based Invocation:** Web services in SOA are discovered, invoked and paid for the usage. Hence, consumers subscribe core assets web services, rather than purchasing and owning them permanently [12].

3.2.2 Quality attributes of reusability

It is possible and feasible for an organization to use core assets as reusable web services, so reusability model are derived from key features as shown in figure (3.3), to evaluate the reusability of core assets web service using the following attributes:

1- Functional commonality

Functional commonality measures commonalities of functions among product line scope applications. The rationale for defining this attributes that the functions developed by providers should be common to all application defined in product line requirement specification. If core assets web service provides number of superior functionality, it will not be extensively subscribed by consumer and will not be extensively reused unless those functions are common among all applications [2].

2- Nonfunctional commonality

Nonfunctional commonality measures commonalities of nonfunctional requirements which it provided by core assets web service to family members define in product line scope. Quality attributes are identified by determining common nonfunctional requirements which is the main driver of the product line architectures, so if nonfunctional requirements are common, the product line architectures will be common to all family members. The rational for defining this attributes that the reusing of architecture is the most important concept in product line engineering comparison with other reusable approaches [2].

3- Modularity

Modularity measures the extent of core assets to provide separate functionality without reliance on other core assets. There for to increase easily reusability of functionality, the core assets must be modularized, that means reducing coupling and increasing in cohesion. This feature of core assets can be achieved by separation of concern about service interface and implement. The rational for defining this attributes that if the core assets is depend on other core assets, the consumer must subscribe all core assets have relationship with it, Also the load will be on consumer by increasing the complexity of composition. Therefore, modularity is a key factor which affects the reusability of a service [12].

4- **Variability richness**

Variability richness measures if core assets in web service comprehend all variability in product line scope, if the variability is sufficiently captured in the core assets web service, then larger number of consumer can reuse the core assets web service by selecting better scope and identifying specific variability for consumer application. The rational for defining this attributes that if we define suitable number of variability the core assets web service will be more reusable but if we define restricted number of variability then it will reduce the reusability so the big challenge waylays in defining suitable number of variability that motive reusing mechanism by huge number of family member [15]. In addition, if the boundary of variability is confined, then it will be reused by a restricted number of applications [2].

5- **Applicability**

Applicability measures the capability of family member to apply core assets web service. So we must determine all applications which can reuse these core assets obviously. The rational for defining this attributes that the large number of family member applying the core assets web service attests increasing in reusability rate. This attribute derived from functional commonality, nonfunctional commonality and variability richness. It can’t be apply if consumer doesn’t find it expectation correctly and in our case the consumer will found all that which they need because it define previously before implementing the core assets web service [2].

6- **Standard conformance**

Standard conformance measures the extent of conforming international industry standards that are widely accepted and used, so that this attribute help in understanding core assets web service which lead to increase in reusability. The rational for defining this attributes is that large number of family member will composite core assets web service effectively without complication due to applying standard. By applying standard, specification will be widespread acceptance which it makes the communication and understanding between family member and core assets web service are possible and available, all that goes towards increasing the concept of reusability. This means that the core assets web service with high standard conformance provides the high probability of high reusability [12].

7- **Tailorability**
Tailorability measures the capability of core assets web service to provide functionality which it fits all or meet all family member application mention in PL scope. The tailoring of core assets web service functionality is very necessary because it encourage this web service to be widely used by family member which led to increasing reusability. The rational for defining this attributes that the subscribing for core assets service is available for every family member [2].

8- **Component replicability**
Component replicability measures possibility of replace embedding component by COTS or in-house component without changing in product line architecture; the change in product line architecture by replacing other component will change core assets web service design so invocation process may cause some unexpected problems. The rational for defining this attributes is that the core assets web service is embedding components. As there for the important for this attribute shown clearly when we find new application not in PL scope and we can replace some core assets by suitable replaced component [2].

9- **Discoverability**
Discoverability measure the extent of family member to find core assets web service which they are expecting or looking for easily and correctly. Family member can search for any core assets web service depend on their specifications and the query return the ideal results about their searching. More over the process of discovering the ideal services need pre preparation from service provider, as well as they must define core assets web service specifications which should be specified in easily understandable manner firstly. Then it can be feasibly and correctly comprehended by family member. The rational for defining this attributes is that if service provider well-define core assets web service specifications then the family member can more easily understand the web service specifications and provide a high possibility to be discover duo to demonstration all necessity to family member, so that it will provide the high possibility of reuse by family member [12].

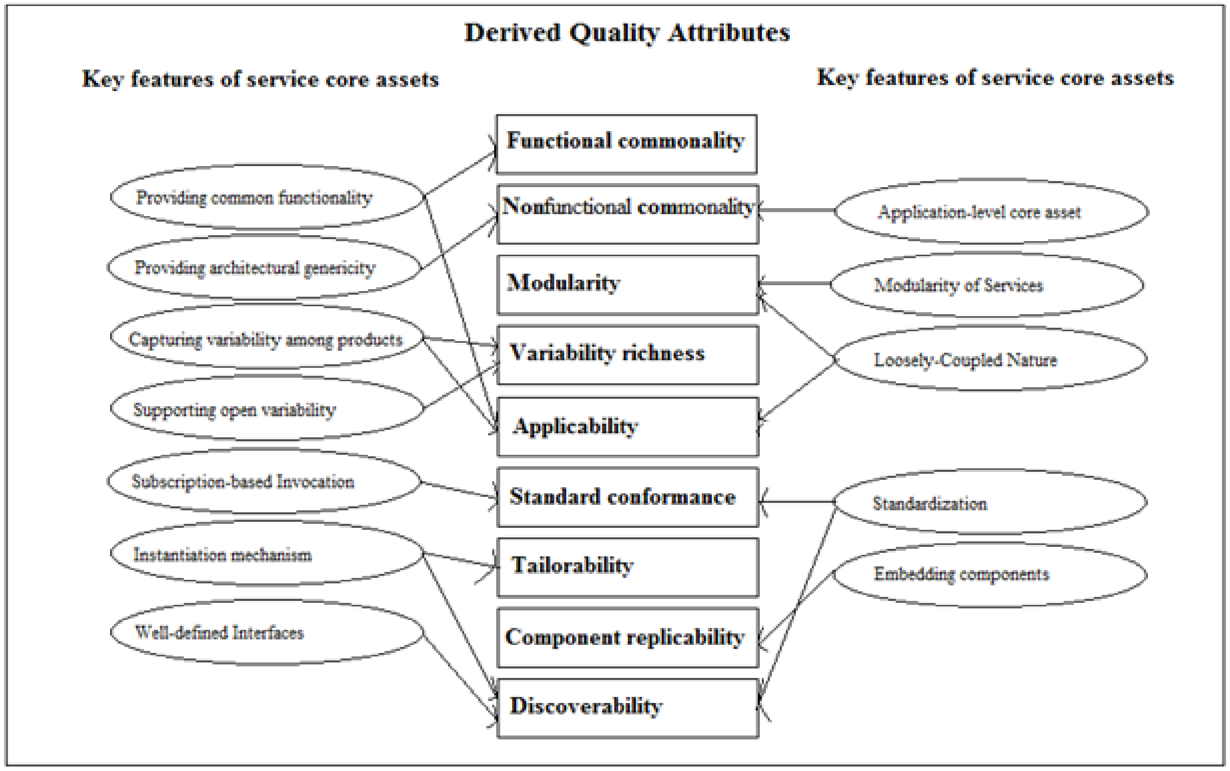

Derived Quality Attributes
Key features of service core assets
Key features of service core assets
Functional commonality
Nonfunctional commonality
Modularity
Variability richness
Applicability
Standard conformance
Tailorability
Component replicability
Discoverability
Providing common functionality
Providing architectural genericity
Capturing variability among products
Supporting open variability
Subscription-based Invocation
Instantiation mechanism
Well-defined Interfaces
Application-level core asset
Modularity of Services
Loosely-Coupled Nature
Standardization
Embedding components

Figure (3.3) Mapping between key features and quality attributes of reusability.

3.2.3 Metrics for Reusability of service core assets oriented

In this section, we define metrics for each attribute in terms of metric description, formula, value range, and relevant interpretations. Figure (3.5) shows the corresponding metrics for quality attributes:

1- **Functional commonality**

This attributes is measure by Functional Coverage (FC) metric, FC measures the average of commonality for each functional feature in core assets web service, commonality of each feature can be measure by calculating the degree of family member using each functional feature [2]. This can be computed as;

$$fc = \left(\sum_{i=1}^{n} \frac{number\ of\ application\ using\ ith\ feature}{total\ number\ of application\ in\ product\ line\ scope} \right) / \, n$$ [2]

Where n is total number of functional feature.

The range degree of FC between 0 and 1, 1 mean the all functional feature are common among all application in product line scope and 0 mean there is no common feature are shareable.

2- Nonfunctional commonality

This attributes measure by Architectural Commonality (AC) and Nonfunctional Coverage (NC) metrics, where AC measures nonfunctional requirement that are tackled by PL architecture and NC measures the average of non-functional commonness feature that are not involve in product line architecture [2].

$$Ac = \frac{number\ of\ application\ sharing\ product\ line\ architecure}{total\ number\ of\ application\ in\ product\ line\ scope}$$ [2]

The range degree of AC between 0 and 1, hence 1 is mean all of application sharing the PL architecture and 0 there is no application share the PL architecture.

$$NC = \left(\sum_{i=1}^{n} \frac{number\ of\ application\ using\ ith\ nonfunctional\ feature}{total\ number\ of\ application\ in\ product\ line\ scope}\right) / n$$ [2]

Where n is the number of non-functional feature that are not handled by PL architecture.

The range of NC between 0 and 1, 1 means all non-functional features that are not handled are common among consumer member [2].

From the above two metrics we can calculate Non-Functional commonality (NFC) as [2]:

$$NFC = W_{AC} \cdot AC + W_{NC} \cdot NC$$ [2]

Where W_{AC} and W_{NC} are the weights for each metric which the summation of both 1.

The value of each weight is getting by considering the ratio of non-functional that are handled and not handled by PL architecture.

The value of NFC between 0 and 1, 1 means the application has large applicability; hence it used PL architecture and non-functional feature [2].

3- Modularity

This attributes measure by Modularity (MD) metric, MD measures the degree to which a core assets is independent of others core assets [12].

$$MD = 1 - \left(\frac{num_{of\ core\ asset\ that\ are\ dependent\ on\ other\ core\ asset\ in\ a\ service}}{num_{of\ total\ core\ asset\ in\ service\ core\ asset}}\right)$$ [12]

The range of MD between 0 and 1, the value 1 means all core asset with in core asset web service are independent; so the core asset web service are self-contain.

4- **Variability richness**

This attributes measure by Coverage Variability (CV) metric, where CV is measures the rate of variation point realized in core asset web service and variation point in product line scope [2].

$$CV = \left(\frac{number\ of\ variation\ point\ realized\ in\ core\ asset\ web\ service}{number\ of\ variation\ point\ included\ in\ product\ line\ scope}\right)$$ [2]

Where the denominator is the number of variation point identified from SRS and included in the scope of product line.

The range of CV between 0 and 1, hence the value 1 means all variations point in PL scope are realized in core asset; that a pones which let a lot of family member consumer to reuse this core asset.

5- **Applicability**

This attributes measure by Cumulative Applicability (CA) metric, CA is measures the ability of core assets web service to develop various application by family member. This attributes is composite metric which depend on FC, NFC and CV metrics [2], therefore it can compute as;

$$CA = W_{FC} \cdot FC + W_{NFC} \cdot NFC + W_{CV} \cdot CV$$ [2]

Where $W_{FC},\ W_{NFC}\ and\ W_{VC}$ are the weights for each metric which the summation of them 1.

The value of weight for each metrics calculated as it mention in [2]. The author divides the feature to four categories as shown in figure (3.4), after that he calculate the weight as following [2]:

$W_{FC} + W_{NFC} + W_{CV} = 1$

$W_{FC}: W_{NFC}: W_{CV} = (a+b):(c+d):(b+d)$

Let x be the multiplicative constant for calculating $W_{FC}, W_{NFC}\ and\ W_{CV}$, and then each weight has the value of $(a+b)x, (c+d)x\ and\ (b+d)x$. Through these formulas x is $1/(a+2b+c+2d)$. Therefore the value of weights as following:

$W_{FC} \ = (a+b)/(a+2b+c+2d)$

$W_{NFC} = (c+d)/(a+2b+c+2d)$

$W_{CV} \ = (b+d)/(a+2b+c+2d)$

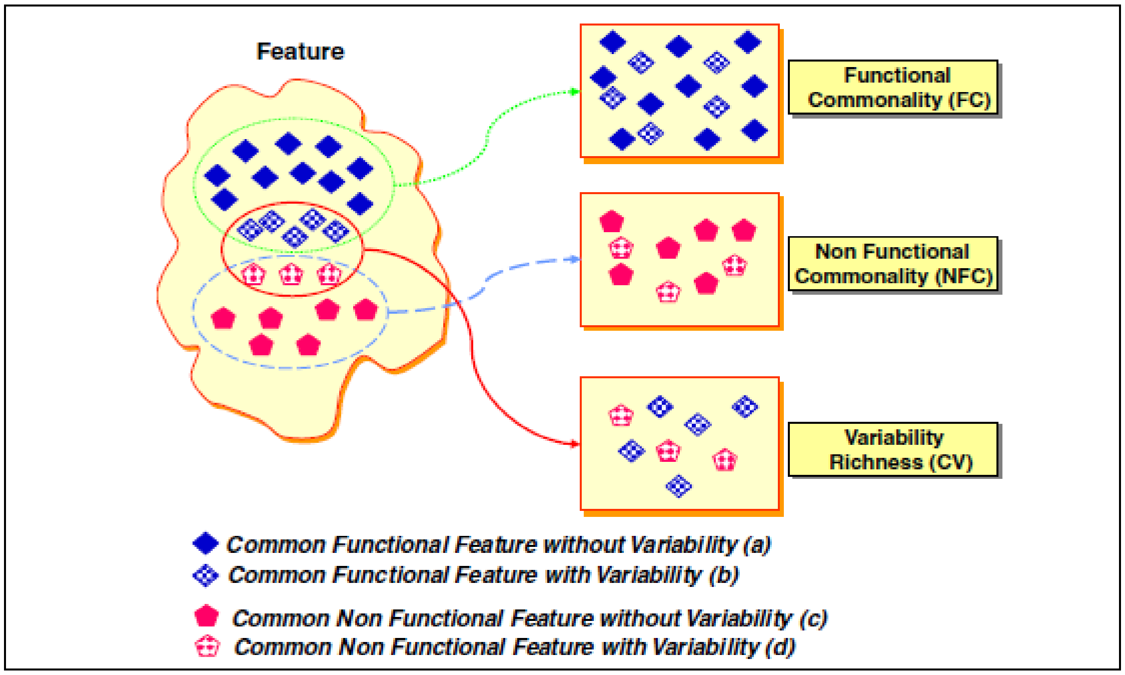

Feature
Functional Commonality (FC)
Non Functional Commonality (NFC)
Variability Richness (CV)
Common Functional Feature without Variability (a)
Common Functional Feature with Variability (b)
Common Non Functional Feature without Variability (c)
Common Non Functional Feature with Variability (d)

Figure (3.4) Feature dealt within FC, NFC and CV [2].

The range of CA between 0 and 1, therefore 1 mean lager number of application will be developed by family member from reusing service core asset.

6- **Standard conformance**

This attributes measure by Standard Conformability (SC) metric, SC measures degree of core assets web service conformity to relevant standards. There are two types of standards, the first one is mandatory standard which the core assets web service must conform to, and the second type is optional standard which core assets web service should conform to [12]. So we can calculate SC as:

$$SC = \left(\frac{W_{Mandtory\ Std} \times Num_{Conform\ to\ mandotory\ std}}{Num_{total\ mandotory\ std}}\right) + \left(\frac{W_{Optional\ std} \times Num_{conform\ \ to\ optional\ std}}{Num_{total\ optional\ std}}\right)$$ [12]

Where $Num_{Conform\ to\ mandotory\ std}$ is the number of core assets web service must conform to standard, $Num_{Conform\ to\ optional\ std}$ is the number of core assets web service should conform to standard, $Num_{total\ mandotory\ std}$ is the total number of mandatory standard evaluated by quality evaluator, $Num_{total\ optional\ std}$ is the total number of optional standard evaluated by quality evaluator, $W_{mandotory\ std}$ is the weight for number of mandatory service core asset and $W_{Optional\ std}$ is the number of optional service core asset.

The value of summation for weight is 1. There for the range of SC between 0 and 1. 1 means the service core asset is conforming to all relevant standards.

7- Tailorability

This attribute measures by Tailorability (TL) metric, which is measure the number of variation point (VP) that can be effectively tailored, that mean the number of valid VP after tailoring. TL is composite metrics so; we calculated from Effectiveness Tailoring (ET) metric, Tailorability of closed variability (TC) metric and Tailorability of open variability (TO) metric. The first is ET metrics which is measure how many VP are resolved efficiently before subscribing mechanism [2] as following:

$$ET = \left(\frac{number\ of\ effectively\ resolvable\ varaition\ points}{total\ number\ of\ Varaition\ points}\right)$$ [2]

Where variation point are effectively resolvable when the subscribe mechanism for each variation point is appropriately and efficiently defined considering the constraint of the variation point.

The range of this metric is difficult to quantitatively measure effectively resolvable VP, so we used check list for deciding if each VP is effective to resolve or not. We define checklists considering type and scope of each VP such Binary, optional, alternative and open variability [2].

The second is TC metric which is measure the degree of subscribing closed VP in Decision Resolution Model (DRM) without side effect or fault after subscribing mechanism. The closed VPs are known and every one of them has effect on core asset web service to produce different application [2], so we calculate it as:

$$TC = \frac{\sum_{i=1}^{n}(number\ of\ valid\ varient\ for\ VP_i)}{\sum_{i=1}^{n}(total\ number\ of\ varients\ for\ VP_i)}$$ [2]

Where VP_i is *i*th closed variant point among effective resolvable variation point.

The range of TC between 0 and 1, the higher value of TC 1 means larger number of clothed variation points are validity resolved.

The last metric is TO which is measure the degree of subscribing open VP in DRM without side effect or fault after subscribing mechanism. The open VPs are unknown [2], so we can calculate it as:

$$TO = \left(\frac{number\ of\ valid\ open\ varaition\ points}{total\ number\ of\ open\ varaition\ points}\right)$$ [2]

Where the denominator is the number of open variation point among effectively evolvable variation point.

The range of TO between 0 and 1, and 1 value indicates larger number of open variation points are validity resolves.

By using the preceding metrics, the final value of TL measure as [2]:

$$TL = ET \cdot (W_{TC}.TC + W_{TO}.TO) \quad [2]$$

Where W_{TC} ratio of closed variation is points among the total effectively evolvable variation points and W_{TO} is ration of open variation point among the total effectively evolvable variation points, the sum of these weights is 1.

The range of TL is between 0 and 1, hence 1 means tailoring mechanism for each VP is effectively defined and each VP can be resolved without outside effect.

8- Component replicability

This attribute measure by Component compliance (CC) metric, it measures the ability to replace the core asset components in core asset web service without complication. These components may be from COTS or pre develop in-house [2].

$$CC = \left(\frac{number\ of\ replacable\ core\ asset\ components}{total\ number\ of\ core\ asset\ components\ in\ core\ asset\ web\ service}\right) \quad [2]$$

The range of CC between 0 and 1, as well as 1 means all core asset components are replaced without complication.

9- Discoverability

This attributes is measure by Discoverability (DC) which it measures the ability to discover core assets web service easily and correctly by family member. For easily and correctly discovering DC is measure by two metric, the first is Syntactic Completeness of Service Specification (SynCSS) and the second is Semantic Completeness of Service Specification (SemCSS). The completeness means the number of elements that are open-faced to family member because they are well specified in core assets web service specification [12].

SynCSS measures how many syntactic elements are well specified in the core assets web service as following [12]:

$$SynCSS = \left(\frac{number\ of\ well\ described\ syntatic\ element}{total\ number\ of\ syntatic\ element}\right) \quad [12]$$

Where the numerator is element that exposes to family member and the denominator is total number of syntactic element in core assest web service specification.

The range of SynCSS between 0 and 1, and 1 indicate contents of all tags related to signature of core assets web service operations.

SemCSS measures how many semantic elements are well specified in the core assets web service as following [12]:

$$SemCSS = \left(\frac{number\ of\ well\ described\ semantic\ element}{total\ number\ of\ semantic\ element}\right)$$ [12]

Where the numerator is element that exposes to family member consumer and the denominator is total number of semantic element in core assets web service specification.

The range of SemCSS between 0 and 1, and 1means the content of semantic information for core assets web service operation are defined in human language. Finally we can combine the DC by combine the two above metrics [12] as following:

$$DC = W_{Syn} \cdot SynCSS + W_{Sem} \cdot SemCSS$$ [12]

Where W_{Syn} is the weight for SynCSS and W_{Sem} is the weight for SemCSS. The summation of the weights is1.

The higher value of DC indicates a better discoverability.

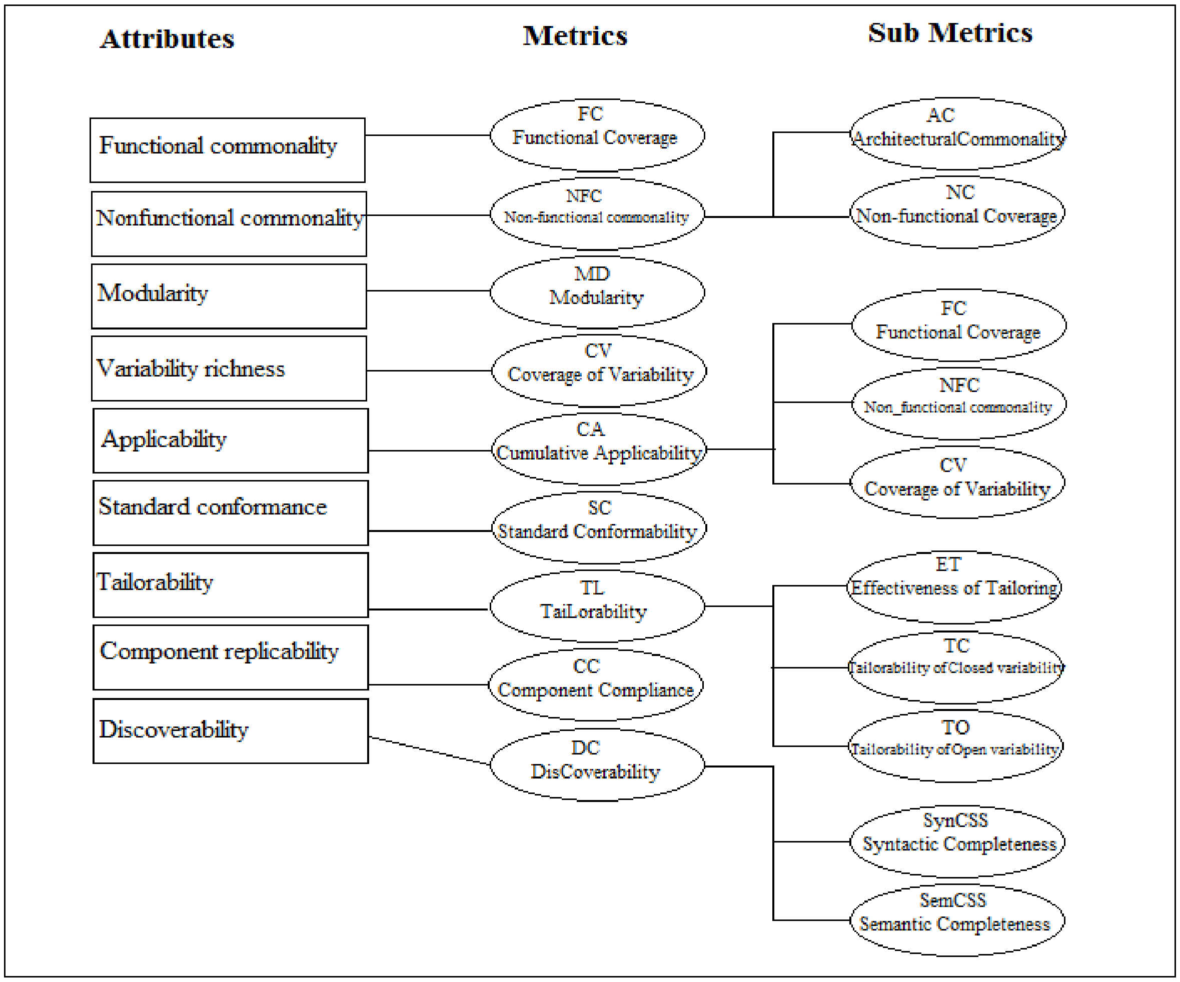
10- Reusability

The final value of reusability can be calculated by Reusability (RE) metric from the nine metrics mention above as following:

$$RE = W_{FC} \cdot FC + W_{NFC} \cdot NFC + W_{MD} \cdot MD + W_{CV} \cdot CV + W_{SC} \cdot SC + W_{TL} \cdot TL + W_{CC} \cdot CC + W_{DC} \cdot DC$$

Where W_{FC}, W_{NFC}, W_{MD}, W_{CV},W_{SC}, W_{TL}, W_{CC} and W_{DC} are weight for each metric. And the summation for this weight equal 1.

The last range of total reusability is from 0 to 1, 1 indicates better reusability.



Attributes
Metrics
Sub Metrics
Functional commonality
Nonfunctional commonality
Modularity
Variability richness
Applicability
Standard conformance
Tailorability
Component replicability
Discoverability
FC Functional Coverage
NFC Non-functional commonality
MD Modularity
CV Coverage of Variability
CA Cumulative Applicability
SC Standard Conformability
TL TaiLorability
CC Component Compliance
DC DisCoverability
AC ArchitecturalCommonality
NC Non-functional Coverage
FC Functional Coverage
NFC Non_functional commonality
CV Coverage of Variability
ET Effectiveness of Tailoring
TC Tailorability of Closed variability
TO Tailorability of Open variability
SynCSS Syntactic Completeness
SemCSS Semantic Completeness

Figure (3.5) Mapping between quality attributes of reusability and metrics.

3.2.4 Build the reusability framework

This framework is built to help any developer to assess the reusability of core assets web service before it will be instantiate by consumers. Also to help them improve weakness area by defining the lowest framework attributes value. So to apply this framework we should be sure that the proposed system which we want measure its reusability using software product line and service oriented architecture approaches, then apply the following step as shown in figure (3.6):

1- Define artifact that will be used in measuring, artifacts are define as high-level model which it is not relevant to specific programing language. Figure (3.7) show the corresponding artifacts for each framework attributes and metrics.

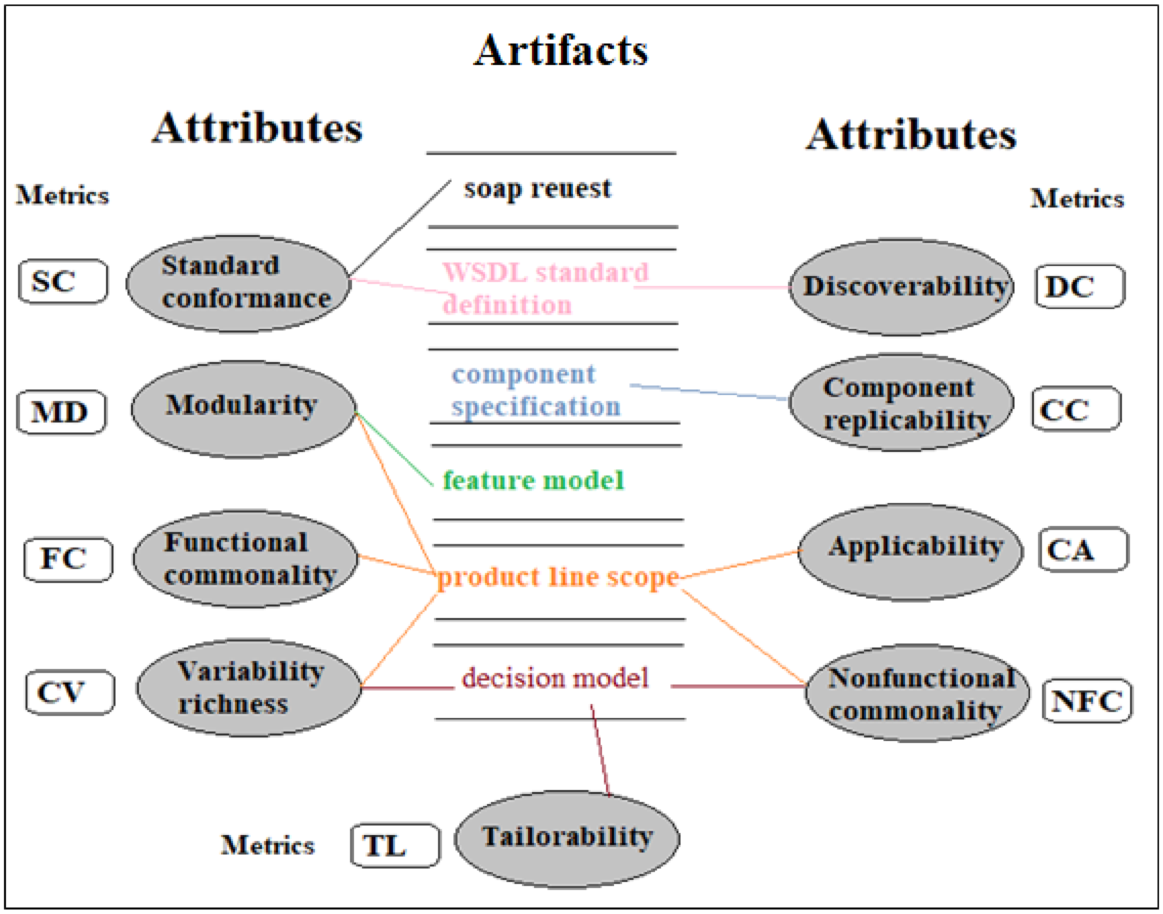


Figure (3.7) Mapping between quality attributes, metrics and artifacts

2- Select the attributes which support the reusability, the attributes aren't selected randomly, but it is derived from core assets and web services characteristic which serves reusability.
3- Apply the predetermine metric for each attributes. The metrics have formulas which computed from direct artifact, each metric is destined to specific attribute. We have two types of metrics direct metrics and complex metric, which is depend on other metrics.
4- Calculate the final result. The final value of reusability is computed according to reusability a formula which is calculated from other metrics.
5- Discuss the result. In the discussion we represent the result of reusability and its sub attributes. Also, we highlight on lowest value of sub attributes, which effect on final value of the reusability. Therefore we can investigate enhancement chances for reusing core assets web service.
6- Take final decision. According to the judgment from discussion the result step, the developer decides whether to publish this core assets web services or

not. The Interpretation for reusability value is high if it is near 1 and low if it is near 0.

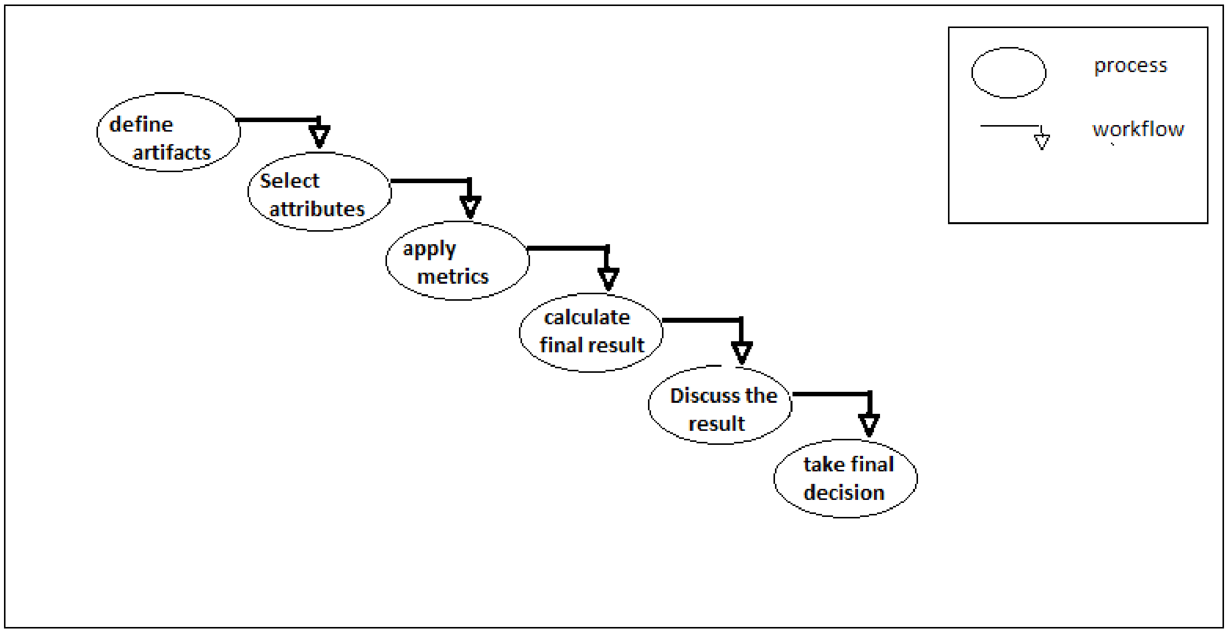
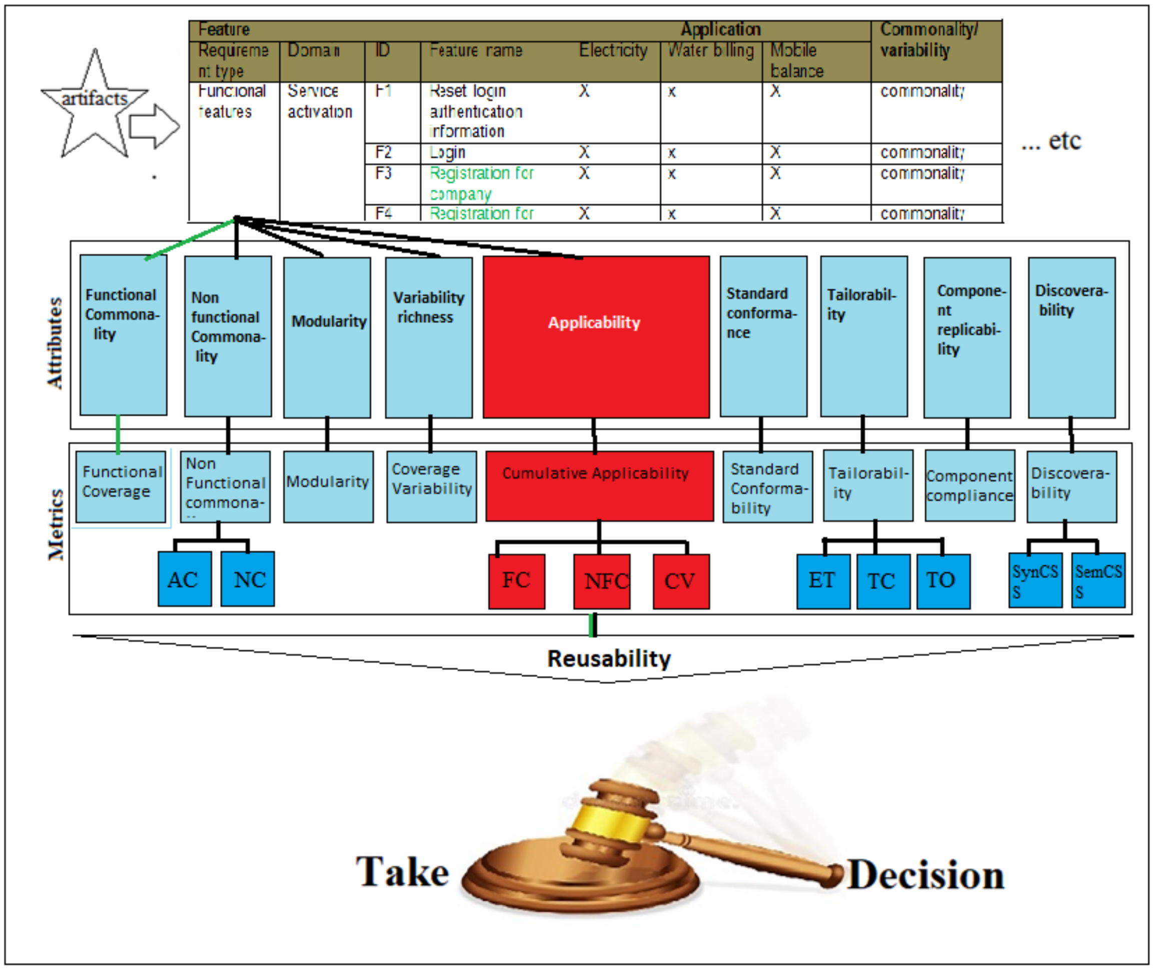


Figure (3.6) Framework steps to evaluate reusability of core assets web service

In order as to applying steps in figure (3.6); we represent framework execution example as shown in figure (3.8) to calculate functional commonality attribute refer to product line scope artifact and using functional coverage metric, the output of this metric contribute in final reusability result. All reusability attributes in the framework model are calculated as same way except applicability attribute which is calculated from three metrics (Functional coverage, Non-functional commonality and Coverage Variability). It is indicating the consumer must find its expectation or their need correctly to apply this core assets web service, in spite of its important that is not used in final value of reusability; because its sub metrics are calculated individually in final reusability result. Therefore, to avoid repeating metrics it's not used in final reusability metrics in the framework model.



Feature				Application			Commonality/ variability
Requirement type	Domain	ID	Feature name	Electricity	Water billing	Mobile balance	
Funcional features	Service activation	F1	Reset login authentication information	X	x	X	commonality
		F2	Login	X	x	X	commonality
		F3	Registration for company	X	x	X	commonality
		F4	Registration for	X	x	X	commonality

Figure (3.8) Example of framework execution

3.2.5 Build the target core asset web service

The Bank Third Party Services (Credit recharge, Electricity buying and Water billing) were built since 2013 as software product line which it used combinations of best software product line methodologies including KOBRA, FODA and FAST, also it developed as mobile app using android and desktop application using java langua ge [19]. Water billing application is merging with electricity buying application now a day, but we use it as separate application in this research [19].

The traditional working mechanism for third party services system that the family members enter to website after authentication and pickup or select core asset features according to the selected application that will generate, after that press download button to keep .exe desktop application or .apk mobile application. Our contribution in working mechanism, instead of using web site; bank third party system will be available as web services with ability to preselect core asset features and consume the response by family member in any platform.

We select third party services because it building in java or android so the reusing for this core asset feature must be work with the same language environment, and this is the point that our research is solving which is make the reusing environment for core asset is dynamic that means not depend on specific language Environment. All that increasing the spread of services which support our main goal the reusability.

Chapter Four

Framework Implementation & Results

4.1 Introduction

This chapter applying reusability framework on the target system and getting the final result of reusability compared with reusability from other model.

4.2 Artifacts Using in Reusability Framework

4.2.1 Product line scope

Is artifact which displays the feature of product and specifies the commonality and variability used for this model as shown in table (4.1).

Table (4.1) illustrate product line scope [19]

Feature				**Application**			**Commonality/ variability**
Requirement type	Domain	ID	Feature name	Electricity	Water billing	Mobile balance	
Functional features	Service activation	F1	Reset login authentication information	X	x	X	commonality
		F2	Login	X	x	X	commonality
		F3	Registration for company	X	x	X	commonality
		F4	Registration for user	X	x	X	commonality
		F5	Activation using bank account number	X	x	X	commonality
		F6	Activation using E-wallet number	X	x	X	commonality
		F7	Sail point user	X	🟥	X	variability
		F8	Normal user	X	x	X	commonality
		F9	Special user	X	x	X	commonality
		F10	Passport number	X	x	🟥	variability
		F11	Nationality number	X	x	X	commonality
		F12	Card ID	X	x	X	commonality
		F13	Preferences	X	x	X	commonality
	Message forwarding engine	F14	Inter transaction messages	X	x	X	commonality
		F15	Send bank account number	X	x	X	commonality
		F16	Receive bank server response	X	x	X	commonality
		F17	Send service information	X	x	X	commonality
		F18	Send generated tag	X	x	X	commonality
		F19	Receive service number	X	🟥	X	variability
		F20	Receive bill number	🟥	x	🟥	variability

		F21	Encrypting forwarded data	x	x	x	commonality
		F22	Decrypting forwarded data	X	x	X	commonality
	Providing service	F23	Request service number	X	🟥	X	variability
		F24	Request bill number	🟥	x	🟥	variability
		F25	Provide service amount category	X	🟥	X	variability
		F26	Receive service result	X	x	X	commonality
		F27	Generate service temporarily tag	X	x	X	commonality
		F28	Selection to paying using bank account	X	x	X	commonality
		F29	Selection to paying using E-wallet number	X	🟥	X	variability
		F30	Confirmation	X	x	X	commonality
	categorization	F31	Give user normal user category	X	x	X	commonality
		F32	Give user sail point category	X	🟥	X	variability
		F33	Give user special user category	X	x	X	commonality
		F34	Purchase by specify the amount of service	X	🟥	X	variability
		F35	Purchase by specify the money amount	X	x	X	commonality
	Communication with third party companies	F36	Creating third party account	X	x	X	commonality
		F37	Sends service number to the user	X	🟥	X	variability
		F38	Send bill number to the user	🟥	x	🟥	variability
		F39	Set minimum and maximum limit to the service	X	🟥	X	variability
		F40	Set advertisement	X	x	X	commonality
		F41	Set offers	X	🟥	X	variability
	Payment	F42	Payment per month or per year	🟥	x	🟥	variability
		F43	User card value	🟥	🟥	X	variability
	Display	F44	Android view	X	x	X	commonality
		F45	Web language	X	x	X	commonality
	reports	F46	Report view	X	x	X	commonality
		F47	Report duration	X	x	X	commonality

	Service gate	F48	Xml vend	X	🟥	🟥	variability
		F49	Phone third party	🟥	🟥	X	variability
		F50	Water third party	🟥	x	🟥	variability
Nonfunctional features	Security	Nfr51	Username and password	X	x	🟥	variability
		Nfr52	Certificates	🟥	🟥	X	variability
	Availability	Nfr53	Health monitor to diagnose the system	X	x	X	commonality

4.2.2 Decision model

Decision model is artifact which represent variation point and related variant as shown in table (4.2), it consist of:

1- Domain-related questions to be answered in developing products.
2- The set of possible answers/decisions to each question.
3- References to the affected artifacts and variation points, or references to the affected decisions (the reference took from requirements specification document).
4- Descriptions of the effect on the assets for each decision, or descriptions of the effects on the answer sets of the affected decisions.

4.2.2.1 Requirements specification Document:

This document specifies the software product line requirements in term of commonality and variability [19]:

1. Service Activation:

1 C1 each user wants to complete the Service Activation to the system must fill up his/her user name.

1 C2 each user wants to activate the application must fill up his/her bank account number.

1 C3 On activation completion system must show message informing user that the process complete.

1 C4 On completion of activation the system should automatically transform user to application page.

1 V5 each user wants to have Service Activation to the system must determine what type is:

1 C5.1 he/she is working in Bank Company.

1 C5.2 he/she is working in the point of sale.

1 C5.3 he/she is working as normal user.

1 V5.4 user can change its type after service activation.

1 V6 each user wants to have service activation to the system must determine the user identity:

1 C6.1 by ID card.

1 C6.2 by National Number.

1 C6.3 by passport number.

1 V7 Service Activation varies according to user type:

1 C7.1 Service Activation for company that represent a user to identify special user.

1 C7.2 Service Activation for normal user with different category.

2. Providing services:

2 C1 each user bank account must be activated.

2 C2 each user must have sufficient bank account balance.

2 C3 each user must determine the type of services.

2.1 Application screen

2.1.1 <u>Water Application screen</u>

2.1.1 C1 user chooses to pay water bill over fixed period of time.

2.1.1 C1.1 system display message to the user if he want to pay per month or per year

2.1.1 C1.2 the user should enter home number

2.1.1 V1.2.1 the system should show message to the user by the amount of bill to be paid per month.

2.1.1 V1.2.2 the system should show message to the user to tell him/her if he/she want to save home number.

2.1.1 V1.2.3 the user chooses to save home number or not.

2.1.2 <u>Electricity application screen</u>

2.1.2 C1 the user should enter electricity counter number machine.

2.1.2 V2 the user should identify the provided power by money or by kilo byte.

2.1.2 C2.1 system should display a message to the user if he/she want to save the user counter number machine or not.

2.1.2 C2.2 user chooses to save electricity counter number machine.

2.1.3 <u>Balance recharge application screen</u>

2.1.3 C1 the user should enter phone number.

2.1.3 V2 the system display a message to the user if he/she want to save the phone number or not.

2.1.3 C3 display categories of available credit cards for the user to choose from them.

2.1.3 C4 the user should select the card value.

2.1.3 V5 user selection may be from the given range of charge.

2.2 Payment

2.2 C1 the user should pay for any service by his/her Tag number.

2.2 C2 the user must have sufficient account when she/he requests a service.

2.2 V3 user should be granted to service or not according to validity of tag number through Service secret number or Bill amount manipulation.

2.2 V4 payment transaction may be completed either by e-wallet number or account number.

2.2 C4.1 when payment is done by e-wallet user must enter his/her e-wallet number.

2.2 C4.2 when payment is done by account number user must enter his/her account number.

2.3 Notifications

2.3 C1 when user completes his/her transaction successfully the system should send a message to the user to tell him/her the transaction successfully completed.

2.3 C2 when the user has not sufficient account the system should send a message to the user to tell him/her the balance not sufficient to fit the service.

2.3 C3 if user chooses to cancel the operation the system should display verification message.

2.3 V4 the user may confirm to cancel or not.

2.3 C5 if e-wallet number is wrong system must be notified that it’s wrong to try again.

2.3 V5.1 user may try again and enter the number again or not.

2.3 V6 notification may be offers or advertisement.

2.3 V6.1 offers are provided either by bank or Service Company.

2.3 V6.2 advertisements are provided only by Service Company for special occasion.

2.4 Display

2.4V1 display of the system to the user may vary in either android or web language (PHP, Html and CSS).

2.5 Reports

2.5 V1 reports may be visual or literal per fixed period of time.

2.5 V2 user may choose a period of time to be viewed at.

2.6 Message forwarding

2.6 V1 to exchange message between different components of the system the method to do so may be via web service or internally.

2.7 Bank service connection

2.7 V1connection to get service may be done through.

2.7 V1.1 in case of electricity via xml vend.

2.7 V1.2 in case of water through water third party.

2.7 V1.3 in case of credit recharge through phone third party.

3. Security

3 V1 to secure the transformation of data between user, bank and third party over network through

3 V1.1 username and password authentication

3 V1.2 SSL certificates.

4.2.2.2 Decision model artifact:

Table (4.2) illustrate Decision model [19]

Variation point		Decision	Type	Scope	Variant/value	Traceability
ID	VP					
F01	User category	What is the type of the user?	Logic	OR	Normal user	1 V5
					Sail point user	
					Bank user	
F02	Preferences	Is the user can change his/her preferences?	Attribute	Optional	Yes	1 V5.4
					No	
F03	User ID number	Type of identity to identify the user to the system?	Logic	OR	National number	1 V6
					Card id	
					Passport number	

F04	Activation type	What is the type of service activation?	Logic	OR	Company activation	1 V7
					Regular user activation	
F05	Payment style	The system should ask the user to pay per month or per year?	Logic	OR	Month	2.1.1 V1.2.1
					Year	
F06	Service amount categories	The user must identify the provided service amount per money or per company category (kilo meter)	Logic	OR	Company category	2.1.2 V2
					Amount of money	
F07	Card value	User select card value to charge with?	Logic	OR	5 SDG	2.1.3 V5
					10 SDG	
					25 SDG	
					50 SDG	
					100 SDG	
F08	Providing services	Type of valid identification id to grant service for the user?	Logic	Alternative	Service secret number manipulation	2.2 V3
					Bill amount manipulation	
F09	Payment type	What are the tools that the user will use to pay for service?	Logic	OR	E-wallet account	2.2 V4
					Bank account	
F10	Confirmation	The user either confirms the transaction or cancels?	Attribute	Optional	Confirm	2.3 V4
					Cancel	
F11	Notification	What is type of notification will display?	Work flow	Optional	Offer	2.3 V6
					Advertisement	
F12	Display	What is the user view type?	Logic	Alternative	Android	2.4 V1
					Web language	
F13	Report view	What is type of report will be	Logic	Optional	Tabular reports	2.5 V1

		shown to the user?			Visualizing reports	
F14	Report duration	What is the duration of the report the user will generate?	Logic	Optional	Monthly	2.5 V2
					Weekly	
					Daily	
F15	Message forwarding engine	How can send message between user and bank, between bank and third party?	Logic	Alternative	Web services	2.6 V1
					Internally	
F16	Bank service gate	What is the type of gate use to get service from specified third party?	Logic	Alternative	Xml vend	2.7 V1
					Phone third party	
					Water third party	
NF R1 7	Secure transfer	Is there secure transformation of data between user, bank and third party?	Logic	Alternative	Username and password	3 V1
					SSL certificates	

4.2.3 Document Conformance

Is artifact which displays mandatory and optional standard for core assets web service as shown in table (4.3), those standards are divided to three categories core, function-specific and industry-specific [24].

Table (4.3) illustrate Mandatory and Optional standards for core assets web service

Standard name	Type of standard
XML	Mandatory
SOAP	Mandatory
WSDL	Mandatory
WS-Security	Optional
OAI-PMH	Optional
SRW/U	Optional

4.2.4 Check list

Is check list artifact which uses to evaluate variation mechanism, valid variant and open variation points [2] sequentially as shown in table (4.4).

Table (4.4) illustrate Check list evaluation [2]

1- checklist to evaluate if the variation mechanism is correctly designed		
#	**Checkpoints**	**Selection**
1.	Define selection mechanism for optional, Binary and Alternative variation point	☐
2.	Plug-in mechanisms defined for the open variability	☐
3.	External profiles should be effectively designed	☐
4.	The selected variation mechanism should consider the software entity and the binding time	☐
2- checklist for each variant to evaluate if its design is valid or not		
#	**questions**	**selection**
1.	Is the tailoring mechanism for each variant effectively defined?	☐
2.	Is the resolution effect correctly defined?	☐
3.	Is the attached task effectively designed?	☐
4.	Are not there any missing dependencies between variants or variation points?	☐
3- Checklist for valid open variation points		
#	**Checklist**	**selection**
1.	Is the plug-in specification correctly defined?	☐
2.	Is the technique to implement the plug-in specification available?	☐
3.	Is the protocol valid?	☐

4.3 Applying model on core asset service

4.3.1 Functional commonality

To measure this attributes we use functional coverage (FC) metrics which calculate the total number of average application using each functional feature, using product line scope table (4.1) we have 50 functional feature and 3 Application using product line core asset service.

$$\boldsymbol{fc = \left(\sum_{i=1}^{n} \frac{number\ of\ application\ using\ ith\ feature}{total\ number\ of application\ in\ product\ line\ scope} \right) / \ n}$$

So the total range degree of FC is calculated as following in table (4.5):

Table (4.5) illustrate the result of FC

#	FEATURE ID (ID)	NUMBER OF APPLICATION USING ITH FEATURE (T)	TOTAL NUMBER OF APPLICATION USING PRODUCT LINE (M)	T/M
1.	F1	3	3	1
2.	F2	3	3	1
3.	F3	3	3	1
4.	F4	3	3	1
5.	F5	3	3	1
6.	F6	3	3	1
7.	F7	2	3	0.67
8.	F8	3	3	1
9.	F9	3	3	1
10.	F10	2	3	0.67
11.	F11	3	3	1
12.	F12	3	3	1
13.	F13	3	3	1

14.	F14	3	3	1
15.	F15	3	3	1
16.	F16	3	3	1
17.	F17	3	3	1
18.	F18	3	3	1
19.	F19	2	3	0.67
20.	F20	1	3	0.33
21.	F21	3	3	1
22.	F22	3	3	1
23.	F23	2	3	0.67
24.	F24	1	3	0.33
25.	F25	2	3	0.67
26.	F26	3	3	1
27.	F27	3	3	1
28.	F28	3	3	1
29.	F29	2	3	0.67
30.	F30	3	3	1
31.	F31	3	3	1
32.	F32	2	3	0.67
33.	F33	3	3	1
34.	F34	2	3	0.67
35.	F35	3	3	1
36.	F36	3	3	1
37.	F37	2	3	0.67
38.	F38	1	3	0.33
39.	F39	2	3	0.67
40.	F40	3	3	1
41.	F41	2	3	0.67

42.	F42	1	3	0.33
43.	F43	1	3	0.33
44.	F44	3	3	1
45.	F45	3	3	1
46.	F46	3	3	1
47.	F47	3	3	1
48.	F48	1	3	0.33
49.	F49	1	3	0.33
50.	F50	1	3	0.33

The total summation of (T/M) = (31*1 + 11*0.67 +8*0.33) = 41.01

Fc = 41.01/50 = 0.82

The rate of FC is 82%.

4.3.2 Non Functional commonality

To measure this attributes we use Non-functional commonality (NFC) metric which is measure by two sub metrics, Architectural Commonality (AC) and Nonfunctional Coverage (NC) metrics, where AC calculate the average of total number application sharing product line architecture and NC calculate the total number of average application using each nonfunctional feature.

$$\boldsymbol{Ac = \frac{number\ of\ application\ sharing\ product\ line\ architecure}{total\ number\ of\ application\ in\ product\ line\ scope}}$$

There are 3 nonfunctional requirements, all of them relevant to the architecture and there is no conflict between them and it used by three members, there for the value of AC is: (3/3) = 1

The rate of AC is 100%.

The total range degree of NC is calculated as following:

$$\boldsymbol{NC = \left(\sum_{i=1}^{n} \frac{number\ of\ application\ using\ ith\ nonfunctional\ feature}{total\ number\ of\ application\ in\ product\ line\ scope}\right) / n}$$

There is no nonfunctional feature irrelevant to architecture so NC = 0

The rate of NC is 0%.

From AC and NC, the value of NFC calculates as following:

$$NFC = W_{AC} \cdot AC + W_{NC} \cdot NC$$

The weight of $W_{AC} is\ nonfunctional\ feature\ relevent\ to\ PL\ architecture$(3/3) =1, the weight of W_{NC} will be 0.

NFC = $\mathbf{1 \cdot 1 + 0 = 1}$

The rate of NFC is 100% common among the three members.

4.3.3 Modularity

To measure this attributes we use Modularity (MD) metrics which calculates the independency between core assets in web service. We have 9 core assets depend on other and total number of core assets is 53 from product line scope table (4.1), the relationship between 9 core assets taken from feature model [19].

$$\boldsymbol{MD = 1 - \left(\frac{num_{of\ core\ asset\ that\ are\ dependent\ on\ other\ core\ asset\ in\ web\ service}}{num_{of\ total\ core\ assets\ in\ web\ service}}\right)}$$

$$MD = 1 - \left(\frac{9}{53}\right) ==> 1 - 0.169 = 0.831$$

The rate of MD is 83.1%.

4.3.4 Variability richness

To measure this attributes we use Coverage Variability (CV) metrics which is calculate the average of variation point captured in core assets web service.

$$\boldsymbol{CV = \left(\frac{number\ of\ variation\ point\ realized\ in\ core\ asset\ web\ service}{number\ of\ variation\ point\ included\ in\ product\ line\ scope}\right)}$$

Total number of variation point realized is 17 from decision model table (4.2), and total number of variation from product line scope table (4.1) is 21.

$$\boldsymbol{CV = \left(\frac{17}{21}\right) = 0.809}$$

The rate of CV is 80.9%.

4.3.5 Applicability

To measure this attributes we use Cumulative Applicability (CA) which is calculate the rate of developing applications by consumer. It depends on FC, NFC and CV metrics.

$$\boldsymbol{CA = W_{FC} \cdot FC + W_{NFC} \cdot NFC + W_{CV} \cdot CV}$$

The value of weight for each metrics calculated by defining the product line scope feature in table (4.1) to 4 categories as following:

a: common functional feature without variability = 31

b: common functional feature with variability =19

c: common nonfunctional feature without variability =1

d: common nonfunctional feature with variability =2

$x = 1/(a + 2b + c + 2d)$

$W_{FC} = (a + b)/x$

$W_{FC} = (31 + 19)/\big(31 + 2(19) + 1 + 2(2)\big)$ = 50/74= 0.675

$W_{NFC} = (c + d)/x$

$W_{NFC} = (1 + 2)/\big(31 + 2(19) + 1 + 2(2)\big)$ = 3/74= 0.04

$W_{CV} = (b + d)/x$

$W_{CV} = (19 + 2)/\big(31 + 2(19) + 1 + 2(2)\big)$ = 21/74= 0.283

After defining weights for each metric then calculates CA as following:

$\mathrm{CA} = 0.675 \cdot 0.82 + 0.04 \cdot 1 + 0.283 \cdot 0.809$ = 0.822

The rate of CA is 82.2%.

4.3.6 Standard conformance

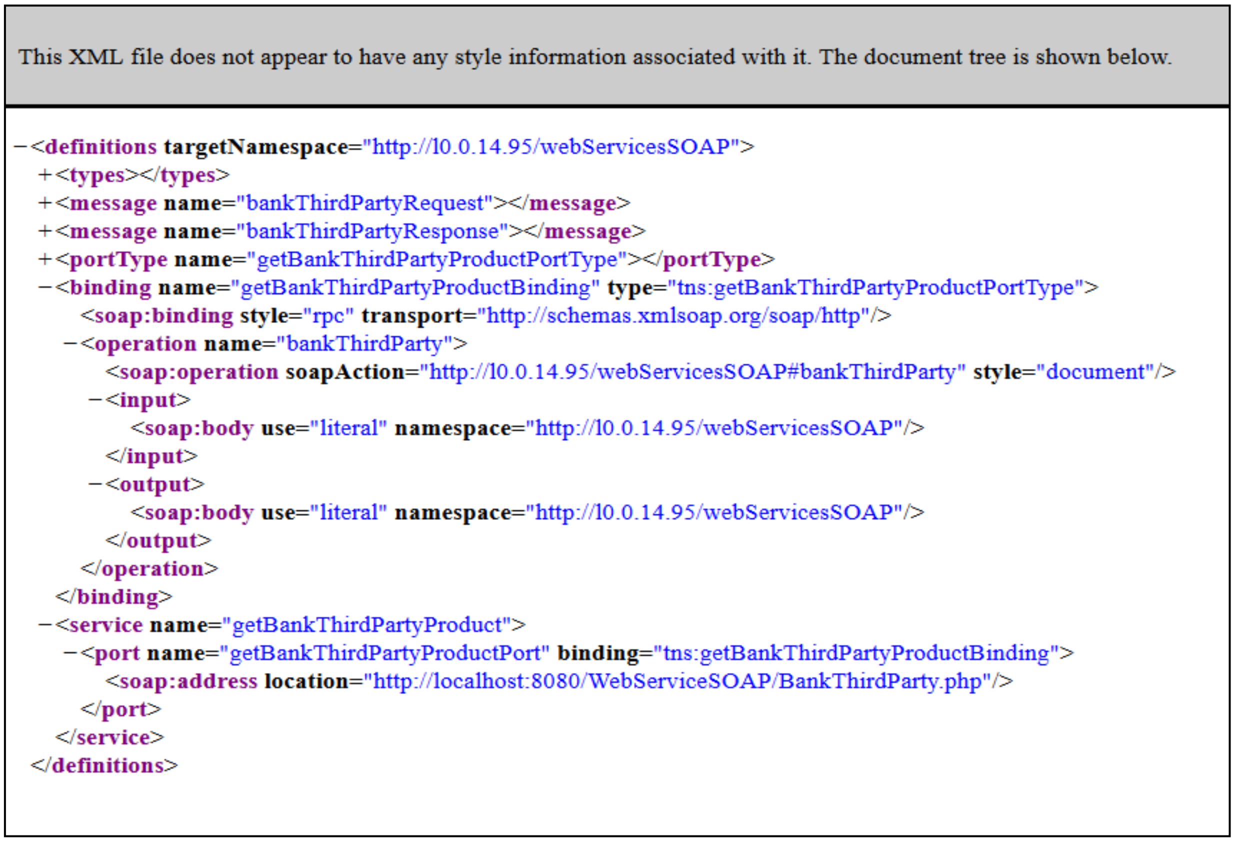
To measure this attributes we use Standard Conformability (SC) metric which is calculate the average of mandatory and optional standard. The core assets web service uses just core standard such as XML, SOAP and WSDL standards.

$$SC = \left(\frac{W_{Mandatory\ Std} \times Num_{Conform\ to\ mandatory\ std}}{Num_{total\ mandatory\ std}}\right) + \left(\frac{W_{Optional\ std} \times Num_{conform\ \ to\ optional\ std}}{Num_{total\ optional\ std}}\right)$$

The core assets web service has one main function and it is implemented by adopting related standard illustrated in Table (4.3). Figure (4.1) illustrate WSDL definition of web service, and figure (4.2) illustrate Soap request.

$$SC = \left(\frac{0.9\times3}{3}\right) + \left(\frac{0.1\times0}{3}\right) = 0.9 + 0 = 0.9$$

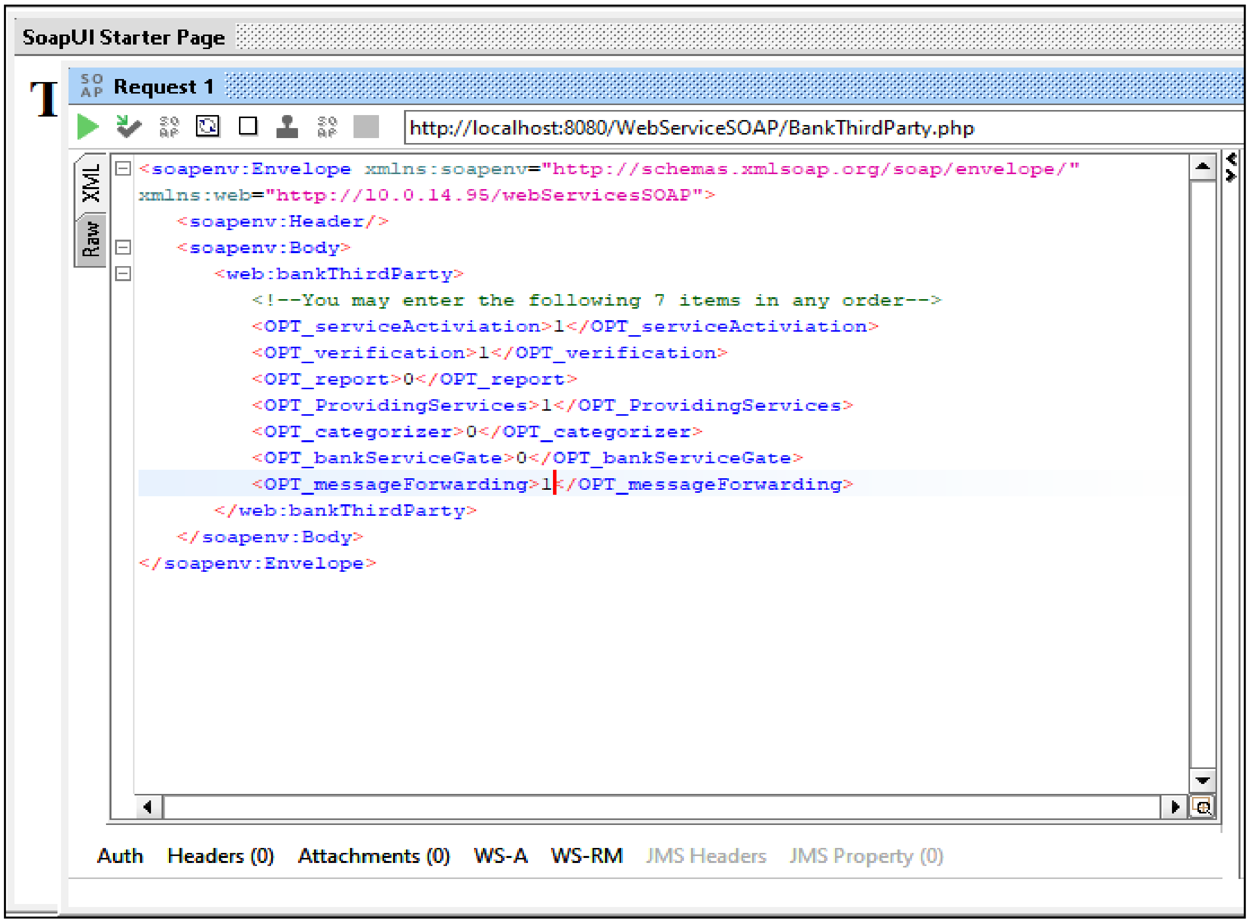
The rate of SC is 90%.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
−<definitions targetNamespace="http://10.0.14.95/webServicesSOAP">
  +<types></types>
  +<message name="bankThirdPartyRequest"></message>
  +<message name="bankThirdPartyResponse"></message>
  +<portType name="getBankThirdPartyProductPortType"></portType>
  −<binding name="getBankThirdPartyProductBinding" type="tns:getBankThirdPartyProductPortType">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    −<operation name="bankThirdParty">
        <soap:operation soapAction="http://10.0.14.95/webServicesSOAP#bankThirdParty" style="document"/>
      −<input>
          <soap:body use="literal" namespace="http://10.0.14.95/webServicesSOAP"/>
        </input>
      −<output>
          <soap:body use="literal" namespace="http://10.0.14.95/webServicesSOAP"/>
        </output>
      </operation>
    </binding>
  −<service name="getBankThirdPartyProduct">
    −<port name="getBankThirdPartyProductPort" binding="tns:getBankThirdPartyProductBinding">
        <soap:address location="http://localhost:8080/WebServiceSOAP/BankThirdParty.php"/>
      </port>
    </service>
  </definitions>
```

Figure (4.1) Illustrate WSDL Definition



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://10.0.14.95/webServicesSOAP">
   <soapenv:Header/>
   <soapenv:Body>
      <web:bankThirdParty>
         <!--You may enter the following 7 items in any order-->
         <OPT_serviceActiviation>1</OPT_serviceActiviation>
         <OPT_verification>1</OPT_verification>
         <OPT_report>0</OPT_report>
         <OPT_ProvidingServices>1</OPT_ProvidingServices>
         <OPT_categorizer>0</OPT_categorizer>
         <OPT_bankServiceGate>0</OPT_bankServiceGate>
         <OPT_messageForwarding>1</OPT_messageForwarding>
      </web:bankThirdParty>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure (4.2) Illustrate soap request

4.3.7 Tailorability

To measure this attributes we use Effectiveness Tailoring (ET) metrics, Tailorability of closed variability (TC) metrics and Tailorability of open variability (TO) metric, where ET calculate the average of resolvable variation point in consuming process, TC calculate the average of valid closed resolvable variation point and TO calculate the average of valid open variation point. We apply the check list 1, 2 and 3 in table (4.4) for ET, TC and TO metric sequentially.

$$ET = \left(\frac{number\ of\ effectively\ resolvable\ varaition\ points}{total\ number\ of\ Varaition\ points}\right)$$

Using check list we got 15 effective resolvable, and the total number of variation point is 17 from decision model.

$ET = \left(\frac{15}{17}\right)$ = 0.882

The rate of ET is 88.2%.

$$TC = \frac{\sum_{i=1}^{n}(number\ of\ valid\ varient\ for\ VP_i)}{\sum_{i=1}^{n}(total\ number\ of\ varients\ for\ VP_i)}$$

Using check list we have 15 valid close variation points and the total number of valid variant is 20, and the total number of variant from variation point in decision model is 37.

$TC = \frac{20}{37}$ = 0.54

The rate of TC is 54%.

$$\boldsymbol{TO = \left(\frac{number\ of\ valid\ open\ varaition\ points}{total\ number\ of\ open\ varaition\ points}\right)}$$

According to there is no open variation point in decision model, TO is equal 0.

The rate of TO is 0%

$$TL = ET \cdot (W_{TC}.TC + W_{TO}.TO)$$

The final weight of $\boldsymbol{W_{TC}}$**and** $\boldsymbol{W_{TO}}$ **are 1 and 0 respectively**

$\boldsymbol{TL = 0.882 \cdot (1 \cdot 0.54 + 0)}$ **= 0.476**

The rate of TL is 47.6%

4.3.8 Component replicability

To measure this attributes we use Component compliance (CC) metric which is calculate the average of replaceable core asset components.

$$\boldsymbol{CC = \left(\frac{number\ of\ replacable\ core\ asset\ components}{total\ number\ of\ core\ asset\ components\ in\ core\ assets\ web\ service}\right)}$$

The total number of component is 9 as mention in component specification [19]. The number of replaceable component is 7 after satisfy the following point [2]:

- If there is a standard or a de facto for component interfaces of the target domain, then the component should conform to the standard.
- If there is no standard or de facto, dependency between the target component and the other components should be low.

- Interface and the component should be clearly separated.
- There should be no or minor side effects after component replacement.
- Specifications of the component should be sufficiently provided.

$\boldsymbol{CC = \left(\frac{7}{9}\right) = 0.777}$

The rate of cc is 77.7%.

4.3.9 Discoverability

To measure this attributes we use Discoverability (DC) metric which is measure by two sub metrics, the first one is Syntactic Completeness of Service Specification (SynCSS) and the second is Semantic Completeness of Service Specification (SemCSS) metrics, where SynCSS calculates the average of total number well describe syntactic element and SemCSS calculates the average of well describe semantic element.

$$\boldsymbol{SynCSS = \left(\frac{number\ of\ well\ described\ syntatic\ element}{total\ number\ of\ syntatic\ element}\right)}$$

According to applying well define standardization such as WSDL, SOAP and XML we can say it is well described as syntactic element, there for the value of SynCSS is 1. Therefor the rate of SynCSS is 100%.

$$\boldsymbol{SemCSS = \left(\frac{number\ of\ well\ described\ semantic\ element}{total\ number\ of\ semantic\ element}\right)}$$

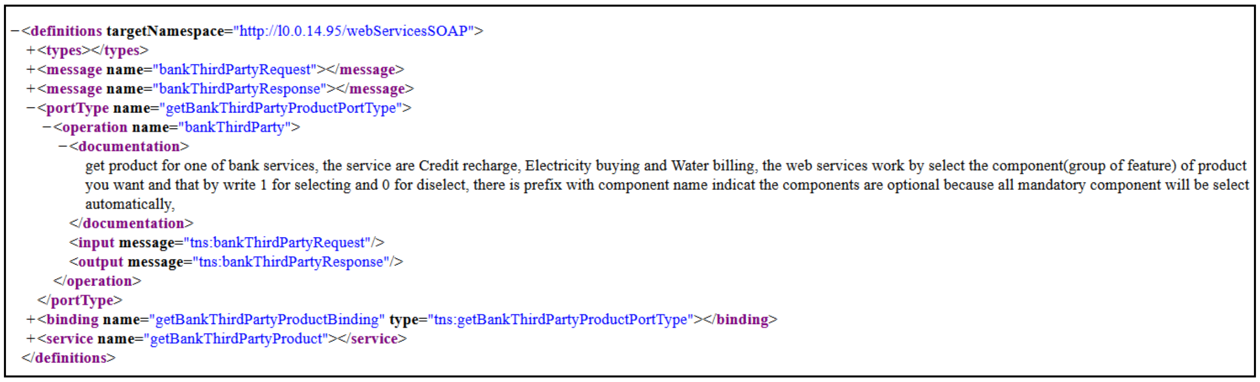
The semantic element mean to describe all web services attributes with human language which it helps in discovery the exact core assets web services as much as good than using just key word, here in our case study it is not provide semantic description but it provide function description as shown in figure (4.3). So the value of SemCSS is 0, and the rate of SemCSS is 0%.

$$\boldsymbol{DC = W_{Syn} \cdot SynCSS + W_{Sem} \cdot SemCSS}$$

To calculate the final value of Discoverability; we have to define weight for each metric, the weight is 0.8 and 0.2 sequentially according to their important.

$\mathrm{DC} = 0.8 \cdot 1 + 0.2 \cdot 0 = 0.8$

The rate of DC is 80%.



```
-<definitions targetNamespace="http://10.0.14.95/webServicesSOAP">
  +<types></types>
  +<message name="bankThirdPartyRequest"></message>
  +<message name="bankThirdPartyResponse"></message>
  -<portType name="getBankThirdPartyProductPortType">
    -<operation name="bankThirdParty">
      -<documentation>
          get product for one of bank services, the service are Credit recharge, Electricity buying and Water billing, the web services work by select the component(group of feature) of product
          you want and that by write 1 for selecting and 0 for diselect, there is prefix with component name indicat the components are optional because all mandatory component will be select
          automatically,
        </documentation>
        <input message="tns:bankThirdPartyRequest"/>
        <output message="tns:bankThirdPartyResponse"/>
      </operation>
    </portType>
  +<binding name="getBankThirdPartyProductBinding" type="tns:getBankThirdPartyProductPortType"></binding>
  +<service name="getBankThirdPartyProduct"></service>
 </definitions>
```

Figure (4.3) illustrate function description in WSDL definition

4.3.10 Reusability

To calculate the final value of reusability (RE) for bank third party core assets web services, we used 6 weights; depend on important of attributes so the total of weights is 1.

$$\boldsymbol{RE = W_{FC} \cdot FC + W_{NFC} \cdot NFC + W_{MD} \cdot MD + W_{CV} \cdot CV + W_{SC} \cdot SC + W_{TL} \cdot TL + W_{CC} \cdot CC + W_{DC} \cdot DC}$$

The weight of metric is taken according to important of attributes and it is direct effect in reusing the core assets. Table (4.6) illustrate the important and according weight for each metric, the range from 6-8 is high, the range from 3-5 is medium and from 0-2 is low. The weight of Applicability is not counted because this metrics is not calculated at the final value of reusability the demonstration that the FC, NFC and CV it will be calculated duple in metric.

Table (4.6) illustrate important and according weight for each attributes

Attributes	Metrics	Priority	Value	Weight
Functional commonality	FC	High	8	0.153
Non Functional commonality	NFC	High	8	0.153
Modularity	MD	High	7	0.134
Variability richness	CV	High	7	0.134
Applicability	CA	-	-	-
Standard conformance	SC	High	6	0.115
Tailorability	TL	High	6	0.115
Component replicability	CC	Medium	5	0.1
Discoverability	DC	Medium	5	0.1
			52	1

$\boldsymbol{RE} = 0.153 \cdot 0.82 + 0.153 \cdot 1 + 0.134 \cdot 0.831 + 0.134 \cdot 0.809 + 0.115 \cdot 0.9 + 0.115 \cdot 0.476 + 0.1 \cdot 0.777 + 0.1 \cdot 0.8 =$

$\boldsymbol{RE} = 0.12546 + 0.153 + 0.12546 + 0.108406 + 0.1035 + 0.05474 + 0.0777 + 0.08 = 0.828266$

The Final rate of RE is 82.8%

After measuring the Reusability, which it was high comparison with the result of reusability came out from using SPL or SOA approaches.

4.4 Result summary

After evaluating the reusability on bank third party services, the result came out from measuring was high comparison with $reusability_{of\ service\ in\ SOA}$[12] and $reusability_{of\ core\ assets\ in\ SPL}$[2] as shown in table (4.7), also reusability is near 1 that means it's high.

Reusability result from SOA model [12] is less than research framework because it isn't include commonality and variability characteristics within its attributes, also reusability result from SPL framework [2] is less than research framework because it isn't considering building reusability attributes for different consumer environments (independent programming language).

For enhancement purpose to achieve higher reusability value more than 82.8%, the variation point (VP) of research case study should be effectively tailored while developing core assets web services; because tailorability attributes is the lowest reusability attributes value, which affect on final value of the reusability.

As mentioned in chapter 1 introduction, the hypothesis was attributes derived from key feature of core assets in software product line and web services in service oriented architecture, which support maximum reuse, those attributes resulting in calculation high reusability value. The hypothesis of this research that stated is achieved. The result of the reusability does not achieve 100%; but we achieved a high percentage which is 82.8%.

Table (4.7) illustrate comparison final value of reusability

Domain	Value	percentage
$Reusability\ of\ research\ Framework$	0.828	82.8%
$Reusability_{of\ service\ in\ SOA}$	0.23	23%
$Reusability_{of\ core\ assets\ in\ SPL}$	0.819	81.9%

Chapter Five

Conclusions and Recommendations

5.1 Conclusion

The output of this research is a framework to evaluate reusability of core assets web service; this framework is used by software product line developer to assess their system.

The framework consists of six steps to apply the framework and getting final result. The final reusability value for this framework is result of collection metrics which is define according to quality attributes derived from characteristic of core assets and web services that is supporting reusability. After getting the final reusability value, then determine the usefulness of proposed core assets web service, according to usefulness result the developer publish core assets web services or not.

This framework was applied on Bank third party services which are credit recharge, Electricity buying and water billing. The final result was obtained which achieve the hypothesis is 82.8% and is considered a high value compared with reusability values used in previous studies.

The success of this framework will permit to leverage the development and reusing of software product line based service oriented architecture systems.

5.2 Recommendations

As a complement to this research and to improve the reusability, there are some recommendations for researchers:

- Describe the framework as xml based as to be machine readable.
- Adding quality attributes which reflect the reusability result for return of investment (ROI) perspective.
- Building comprehensive framework to evaluate reusability from both sides developer and consumer side.

Reference

6 Reference

1. La, Hyun Jung, Jin Sun Her, and Soo Dong Kim. "Framework for evaluating reusability of component-as-a-service (caas)." Principles of Engineering Service-Oriented Systems (PESOS), 2013 ICSE Workshop on. IEEE, 2013.
2. Jin, Ji, Sang, Sung, Soo. "A framework for evaluating reusability of core asset in product line engineering." Information and Software Technology 49.7 (2007): 740-760.
3. Graham, Steve, et al. Building Web services with Java: making sense of XML, SOAP, WSDL, and UDDI. SAMS publishing, 2004.
4. ISO/IEC 26550:2013 Software and systems engineering--Reference model for product line engineering and management.
5. Clements, Paul, and Linda Northrop. Software product lines: practices and patterns. Vol. 3. Reading: Addison-Wesley, 2002.
6. Kim, Soo Dong, Soo Ho Chang, and Chee Won Chang. "A systematic method to instantiate core assets in product line engineering." Software Engineering Conference, 2004. 11th Asia-Pacific. IEEE, 2004.
7. Nagamine, Motoi, Tsuyoshi Nakajima, and Noriyoshi Kuno. "A case study of applying software product line engineering to the air conditioner domain." Proceedings of the 20th International Systems and Software Product Line Conference. ACM, 2016.
8. Marks, Eric A., and Michael Bell. Service-oriented architecture: a planning and implementation guide for business and technology. John Wiley & Sons, 2008.
9. Cerami, Ethan. Web services essentials: distributed applications with XML-RPC, SOAP, UDDI & WSDL. " O'Reilly Media, Inc.", 2002.
10. Balzer, Yvonne. "Improve your SOA project plans." IBM Global Services (2004).
11. Erl, T.: Service-oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, New Jersey, Munich (2005).
12. 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.
13. Cohen, Sholom, and Robert Krut. Managing variation in services in a software product line context. No. CMU/SEI-2010-TN-007. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2010.
14. Montagud, Sonia, Silvia Abrahão, and Emilio Insfran. "A systematic review of quality attributes and measures for software product lines." Software Quality Journal 20.3-4 (2012): 425-486.
15. Park, Shin Young, and Soo Dong Kim. "A systematic method for scoping core assets in product line engineering." null. IEEE, 2005.

16. Mahmood, Ahmad Kamil, and Alan Oxley. "Reusability assessment of open source components for software product lines." International Journal of New Computer Architectures and their Applications (IJNCAA) 1.3 (2011): 519-533.
17. Mahmood, Ahmad Kamil. "A survey and proposed reusability assessment framework for aspect oriented product line core assets." Research and Development (SCOReD), 2009 IEEE Student Conference on. IEEE, 2009.
18. Lee, Jaejoon, and Gerald Kotonya. "Combining service-orientation with product line engineering." IEEE software 27.3 (2010): 35-41
19. M ustafa, samar, ethar and reem. Software product line case study for bank third party services water billing, electricity and credit recharge. Diss. Sudan university of science and technology. 2013..
20. Niu, Nan, et al. "A systems approach to product line requirements reuse." IEEE Systems Journal 8.3 (2014): 827-836.
21. Kim, Jong-Hwan, et al., eds. Robot Intelligence Technology and Applications 2. Berlin: Springer, 2014.
22. Christensson, Per. "Web Service Definition." TechTerms. Sharpened Productions, 10 August 2017. Web. 25 September 2018. <https://techterms.com/definition/web_service>.
23. Booth, David, and Canyang K. Liu. "Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. W3C, June 2007.
24. Wusteman, Judith. "Realising the potential of web services." OCLC Systems & Services: International digital library perspectives 22.1 (2006): 5-9.
25. Istoan, Paul, Jean-Marc Jézéquel, and Gilles Perrouin. Software product lines for creating service-oriented applications. Diss. Master’s thesis, Irisa Rennes Research Institute, 2009.
26. Samir, Areeg, and Nagy Ramadan Darwish. "Reusability Quality Attributes and Metrics of SaaS from Perspective of Business and Provider." International Journal of Computer Science and Information Security 14.3 (2016): 295-312.