Jurnal Teknologi

Full Paper

SERVICE-ORIENTED DESIGN MEASUREMENT AND THEORETICAL VALIDATION

Arafat Abdulgader Mohammed Elhag[a,b*], Radziah Mohamad[a]

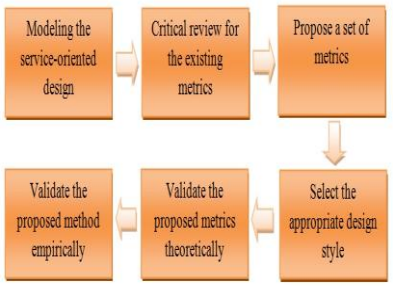
[a]Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor, Malaysia

[b]Department of Computer Science, Faculty of Computer Study, International University of Africa (IUA), 2469 Khartoum, Sudan

Article history
Received
2 February 2015
Received in revised form
8 October 2015
Accepted
12 October 2015

*Corresponding author
arofei@hotmail.com

Graphical abstract



Abstract

As software systems become more and more complex over time, software quality accordingly becomes increasingly important. Service-Oriented Computing (SOC) paradigm is one of the established paradigms used for building and developing flexible, reusable, rapid and low cost software products. Consequently, the use of SOC to develop software systems is increasing. Software quality measurement has considerable importance in the context of SOC since it determines how the quality requirements for composite service should be achieved. As a result, several quality metrics for composite service design were proposed. However, these metrics were constructed based on previous development approaches, give insufficient focus and need modification to be applied to service-oriented systems. Furthermore, the existing metrics do not consider the composite service as building blocks and also they do not consider the indirect relationships. In this paper, a quality measurement for composite service-oriented design is proposed, with the aim of increasing reusability and decreasing the complexity of design. The paper begins with proposing a set of metrics to measure the quality of composite service design. Then, the proposed metrics are validated theoretically to check its usability and applicability for composite service. The results show that the proposed metrics are able to measure the quality of composite service design.

Keywords: Design metrics, coupling; cohesion, complexity, reusability, design metrics, service principles, design properties, theoretical validation

Abstrak

Seiring dengan sistem perisian yang semakin hari menjadi semakin rumit, kualiti perisian juga menjadi semakin penting. Paradigma Pengkomputeran Berorientasikan Servis (SOC) merupakan salah satu daripada paradigma-paradigma yang ada yang digunakan untuk membina dan membangunkan produk-produk perisian yang fleksibel, boleh diguna semula, pantas dan kos rendah. Oleh yang demikian, penggunaan SOC untuk membangunkan sistem perisian ini menjadi semakin meningkat. Penilaian terhadap kualiti sesebuah perisian dianggap penting dalam konteks SOC kerana ia dapat menentukan bagaimana sepatutnya kualiti keperluan untuk servis komposit itu dicapai. Hasilnya, beberapa kualiti metrik untuk reka bentuk servis komposit dicadangkan. Walau bagaimanapun, metrik-metrik yang dihasilkan berdasarkan pendekatan pembangunan sebelum ini adalah kekurangan fokus dan ia memerlukan pengubahsuaian untuk digunakan pada sistem-sistem yang berorientasikan servis. Tambahan pula, metrik-metrik yang telah ada tidak mempertimbangkan servis komposit sebagai blok-blok pembinaan dan hubungan secara tidak langsung juga tidak dipertimbangkan. Kertas ini mencadangkan sebuah ukuran kualiti, dengan tujuan untuk meningkatkan penggunaan semula serta mengurangkan kerumitan reka bentuk. Kertas ini bermula dengan mencadangkan

sebuah set metrik untuk mengukur kualiti servis komposit. Setelah itu, metrik yang dicadangkan telah disahkan secara teori untuk memeriksa kebolehgunaan dan kesesuaian metrik itu sendiri bagi servis komposit. Keputusan menunjukkan metrik yang dicadangkan mampu untuk mengukur kualiti servis komposit.

Kata kunci: Metrik reka bentuk, gandingan, perpaduan, kerumitan, boleh gunapakai, metrik reka bentuk, prinsip-prinsip servis, ciri-ciri reka bentuk, pengesahan teori

1.0 INTRODUCTION

Service-Oriented Computing (SOC) is one of the established paradigms for developing and building the software products [1-3] and it has been applied successfully to develop many types of software systems [4, 5]. A service is an implementation of stateless, self-contained and well defined pieces of functionality, it is published by services provider and can be used by service consumers when building and developing different software systems. The service is designed with interface and operate on published/discovered mode [1]. Initially, software was developed using a procedural paradigm. In the recent past, the procedural paradigm has changed with Object Oriented Computing (OOC) and Component-Based Computing (CBC). Nowadays, a new development paradigm move from previous paradigms to SOC paradigm [6]. However, SOC is quite different from CBC and OOC, because the SOC applying the services as the basic design concept. In contrast, the component used for CBC, and class for OOC [7]. So, service is different than component, because the service functionality is common and not tightly bound to a single client [8].

Nowadays, there are many software applications, which are complex enough, but play a more important role in many areas of our life[9]. To construct and develop these complex applications new development approach is required. SOC has been applied successfully for these applications due to certain benefits which include flexibility, agility, and reusability [4]. As software systems becoming more and more complex over time therefore software quality is also becoming major concern in software development [10, 11]. Software quality is very necessary and essential to many software systems such as the control system, distributed embedded real time system, etc. Quality assurance has a vital role in developing software products because it provides confidence and lowers the risks associated with systems implementation.

SOC promises to deliver the software with high quality due to the advantages such as agility, flexibility, maintainability and reusability [5]. For this purpose, the design element of the SOC has to be designed with many quality attributes. The existing Service Oriented Design (SOD) methodologies describe many quality attributes that comprises, loose coupling, cohesion, autonomy, and reusability. Theses methodologies consider quality attributes as important to achieve SOC goals. But, they do not explain how the design of SOD fulfills these quality attributes and how can measure the quality of service oriented design in term of these quality attributes [12].

SOD has been a very interesting research area under discussion, but there is a potential to consider key principles that guide high-quality design of services. However achieving high-quality design of services in practice is complicated and many service-oriented applications suffer from poor quality and are hard to evolve [13]. The design of software systems in the SOC is made in an ad hoc manner till yet and there is no comprehensive and complete methodology for Service Oriented Architecture (SOA) [1, 4, 14]. Moreover, the success of SOC design depends on the experience and skills of designers. However, measuring the structural properties of service orientation at design level will aid the designers to propose software design fulfills many quality attributes.

Software quality assessment is an important target of software engineering and has a strong impact in the context of service-orientation [3, 11]. There are three approaches to measure the quality of software design, these are; objective approach, subjective approach and hybrid approach. The objective approach measures the structural properties of software systems like coupling, cohesion, autonomy, discoverability, and abstraction. In contrast, the second approach measures the subjective design data and evaluating the quality attributes of services design by measuring quality indicators, that represent the quality attributes and gives value to the current design to help the developers to make a decision about the alternative quality attributes. In addition, the third approach combines first and second approaches. However, the first approach is discussed so much and many authors used this approach and proposed metrics to measure the quality of software. However, further research work is needed for the second and third approach.

The quality of a service-oriented product can be measured when the software product is developed and released. Therefore, assessing and quantifying the quality of the completed software systems will result in the most defined measurements. Disadvantages of this method are discovered defects and faults explored at later stage which will be more costly to fix at the post-production stage. Therefore, several research works have been

established to propose quality measurements that support estimation of software quality of service oriented early in the Software Development Life Cycle (SDLC), particularly at design phase. The key factors in these quality measurements is the structure of service-oriented design properties namely; abstraction, autonomy, cohesion, composability, contract, loose coupling, discoverability, reusability and statelessness [1, 15, 16]. Consequently, a large number of metrics have been proposed for measuring and evaluating the structural properties of a SOD [3, 7, 8, 15, 17-22]. These metrics were constructed based on previous development approaches like OOC and CBC development [23]. Similarly, the existing metrics for the service-oriented design are still at a preliminary stage [12]. The metrics development for approaches such as OOC and CBC, do not work well. These metrics are also not good for service-oriented systems without modification due to unique characteristics of service orientation [7]. In most cases, metrics were used to calculate the quality attributes of SOD, such as coupling and cohesion but were unable to establish relationship between the attributes [19, 24]. The existing metrics consider the direct relationships to calculate their values and there is no consideration for indirect relationships [25]. Furthermore, these metrics consider the operations only as building block for service and exclude the composite service which is a service built from other services to decrease the outside interactions [23]. However, the result shows that there is no prominent approach measured all the criteria or design principles used to control the quality of service-oriented at design phase. Therefore, comprehensive and quantitative metrics for estimating the appropriateness of the service design are still missing [8].

The reset of this paper is structured as follows: Section 2.0 introduces the background and related which contains service-oriented design principles and service-oriented measurement. The proposed metrics are introduced in Section 3.0. The design of the basic metrics for measuring the simple attributes of composite service design is presented in Section 3.1, which is followed by the design of derived metrics for composite service design 3.2. The theoretical validation of the proposed metrics is presented in Section 4.0. Section 5.0 provides the discussion of the proposed metrics in order to show its ability for measuring the quality of composite service design and how it fills the gap in the previous metrics. Finally, the conclusion and direction for future works are presented in section 6.0.

2.0 BACKGROUND AND RELATED WORKS

2.1 Service-Oriented Design Principles

The result of service oriented design phase is the basis for the implementation phase [17]. So, the ability to assess the quality of service oriented at the design phase, will aid in early detection of design flaws which will lead to decrease the cost and effort of implementation phase and enhance the quality of the whole system. In traditional software development approaches, such as Procedural and OOC paradigm, the software quality can be predicted. And as a result, improved early in the Software Development Lifecycle (SDLC) using metrics to measure the structural properties of software designs, such as coupling and cohesion [26] [27]. But, the prediction of software quality in service orientation of the initial level in SDLC, exactly in the design phase is seldom discussed [7].The software quality attributes are divided in two types internal and external attributes [28]. There are many internal and external quality attributes identified for the services design that should be fulfilled to achieve the goals related with the service-oriented application. The external quality attributes covers increased flexibility [1, 29-31] reusability [1, 22] and maintainability [7]. The internal quality attributes are design principles which the design of service-oriented application should support and range of these quality attributes covers cohesion, granularity, loose coupling, design size, discoverability, and autonomy [8, 15, 17, 32, 33].

SOC has become a distinct design paradigm which introduces commonly accepted principles that govern the design of software products [1]. To produce a service oriented design with high quality, we must follow a set of service-oriented design principles. As mentioned in [1] there is no common definition of SOA and there is no common description of service-oriented design properties. However, most common set of principle associated with service-orientation are listed in Table 1.

According to [1, 15, 16, 33, 34] there are twelve service-oriented design principles that covers abstraction, autonomy, cohesion, composability, contract, loose coupling, discoverability, reusability, granularity, complexity, design size and statelessness, as showing in Table 1. The following sentences are describing the service-oriented design principles and how they will affect the quality of software [1, 33]:

Coupling as a term means the direct and indirect interaction and dependency between the components of service-oriented systems. Individual services have not direct dependencies between them. The service-oriented principle is to design the system loose coupled.

Table 1 Principle of service-oriented design

#	Principles	[1]	[15]	[16]	[33]	[34]
1	Coupling	✓	✓	✓	✓	×
2	Abstraction	✓	✓	✓	×	✓
3	Reusability	✓	✓	✓	✓	✓
4	Autonomy	✓	✓	×	×	×
5	Composability	✓	✓	✓	×	✓
6	Statelessness	✓	✓	✓	×	×
7	Discoverability	✓	×	✓	×	✓
8	Contract	✓	×	✓	×	×
9	Cohesion	×	✓	✓	✓	×
10	Granularity	×	✓	×	✓	×
11	Complexity	×	×	×	✓	×
12	Design Size	×	×	×	✓	×

Abstraction- the logic part of the service has been hidden from outside of the world. There are three levels of abstraction in service-oriented system that are the operations level, implementation level and the third level is a service. In service-oriented systems, operations (e.g., OO methods) are aggregated into implementation elements (e.g., OO classes) that implements the functionality of a service as exposed through its service interface [7].

Reusability- A service obtains to improve reusability by developing the software using reusable pieces of software functionality called service.

Autonomy- there is many definitions for service autonomy such as self-controlling, self-contained and self-governing [35]. Service autonomy confirms the logics controlled by a service which has a clear exist in boundary.

Composability- a service can be a composite service or atomic service. The composite service is a big service that comprises other atomic services. the services in service-oriented system should be composable and designed with mechanisms to make it easy to compose and control their functionalities.

Granularity- Granularity refers to the number of functionality encapsulated in a service. A coarse grained service would provide numerous different functions and would have a great number of consumers.

Cohesion- For any service-oriented system, cohesion estimates the degree to which the components of the system belong together and the strength of relationship between operations in a service. In other word, cohesion estimates the difficulty of understanding the relationships between services and service operations.

Statelessness- to remain loosely coupled, services do not maintain state information specific to an activity, such as a service request.

Complexity- complexity is the difficulty of understanding the interaction and relationships between the services and services operations. For any service-oriented system, coupling and cohesion used to estimate the degree to which the components of the system belong together and the strength of the relationship between operations in a service.

Contract- the interaction between the services in service-oriented system needs to share only the formal contract that describes the interact services and explains the terms of exchanged information. This means, the services are not need to share all the information during interaction of services together.

Discoverability- the services in service-oriented system should be discoverable and designed with mechanisms to make it easy to discover and understand their descriptions.

2.2 Service-Oriented Design Measurement

Software companies looking for measuring software quality at the design phase before it's going through the implementation and testing phases. Because, discovers the software defects after implementation or released the software to market will be costly more than in the design phase, and requires more effort and spend the development time to find and fix the software fault. In other words, the software defects and errors discovered during testing needs to redesign the software system. Consequently, when the design of software is changing the rest of project effected. There is a direct correlation between discovering and fixing the service oriented system's fault and the time of correcting the software errors.

There are several works in the literature which have tried to propose some metrics for evaluating and measuring the compliance of the service design against some of the design principles. Some of these studies were reviewed as first step to propose a set of metrics for measuring the quality of composite service design. Table 2 shows the metrics proposed to measure service-oriented design. Moreover, the discussion of the existing work on quality measurement for service-oriented design is placed in the following sentences.

In [15, 22] some metrics proposed to assess the reusability of service-oriented. These reusability metrics of service-oriented system have evolved from CB and OO. This clearly indicates that metrics for measuring reusability in service-oriented system are at the early stage and it requires additional work to propose a complete set of reusability metrics for service-oriented systems.

Loose coupling lead to improve the reusability, understandability, flexibility [33] and maintainability [7, 24] of service-oriented design. In these works [7, 24, 25, 36, 37] a set of metrics were appeared to measure the coupling of service-oriented design. These metrics consider only the direct relationships to calculate their values and there is no consideration for indirect relationships. Further research would be focused on proposing a comprehensive set of coupling metrics for service-oriented.

One of the quality attributes as to a service-oriented is cohesion, which is a determining factor for many other desirable features of the software including reusability, understandability [33, 38] and

maintainability [19]. There are many metrics in [8, 15, 18-20, 39] to evaluate the cohesion of service-oriented design. Most of these studies focus their attention on common input and output parameters of service operations in order to estimate the service cohesion. And consider the operations only as building block for a service and do not consider the services.

Complexity is an important aspect for software quality assessment and must be correctly calculated in service-oriented design. A set of metrics are presented in [3, 17] to measure complexity of service-oriented design.

A set of metrics proposed to measure many attributes such as autonomy [17], Composability [15], Discoverability [22], Modularity [20], Granularity [15, 21, 25] but all of this metrics are in initial stage.

Table 2 Quality measurement and its principles

Studies	Abstraction	Autonomy	Cohesion	Complexity	Composability	Coupling	Design size	Discoverability	Granularity	Modularity	Reusability	Statelessness
[36]	×	×	×	×	×	✓	×	×	×	×	×	×
[23]	×	×	×	×	×	✓	×	×	×	×	×	×
[40]	✓	×	✓	×	✓	✓	×	✓	×	×	×	×
[41]	×	×	✓	×	×	×	×	×	×	×	×	×
[24]	×	×	×	×	×	✓	×	×	×	×	×	×
[42]	×	×	×	×	×	✓	✓	×	×	×	×	×
[33]	×	×	✓	✓	×	✓	✓	×	✓	×	×	×
[43]	×	×	×	✓	×	×	×	×	×	×	×	×
[44]	×	×	×	✓	×	✓	×	×	×	×	×	×
[30]	×	×	×	×	✓	✓	×	×	✓	×	×	✓
[3]	×	×	✓	×	×	✓	×	×	✓	×	×	×
[25]	×	×	✓	×	×	✓	×	×	✓	×	×	×
[15]	×	×	✓	×	✓	✓	×	×	✓	×	✓	×
[25]	×	×	×	×	×	✓	×	×	✓	×	×	×
[45]	×	×	×	×	✓	×	×	×	×	×	×	×
[18]	×	×	✓	×	×	×	×	×	×	×	×	×
[46]	×	✓	✓	×	×	✓	×	✓	×	×	×	×
[47]	×	×	✓	×	×	✓	×	×	×	×	×	×
[48]	×	×	×	×	✓	×	×	×	×	×	✓	×
[7]	×	×	×	×	×	✓	×	×	×	×	✓	×
[49]	×	×	✓	×	×	✓	×	×	✓	×	×	×
[20]	×	×	×	×	×	✓	×	×	×	×	×	×
[17]	×	✓	×	✓	×	×	×	×	×	×	×	×
[21]	×	×	✓	×	×	✓	×	×	✓	×	×	×
[20]	×	×	✓	×	×	×	×	×	×	×	×	×
[50]	×	×	×	×	×	×	×	×	×	✓	×	×
[8]	×	×	✓	×	×	×	×	×	×	×	×	×
[22]	×	×	×	×	×	×	×	✓	×	✓	×	×
[51]	×	×	×	×	×	✓	×	×	×	×	×	×
[52]	×	×	×	×	×	✓	×	×	×	×	×	×
[53]	×	×	×	×	×	×	×	×	✓	×	×	×
[54]	×	×	✓	×	×	✓	×	×	×	×	×	×
[32]	×	×	✓	×	×	×	×	×	×	×	×	×

However, the literature of service-oriented design metrics shows the missing of comprehensive measurement to evaluate all service-oriented design properties.

3.0 THE PROPOSED METRICS

Software quality measurement is a necessary target of software engineering and, in addition, has considerable importance in the context of SOD since it determines how the quality requirements for composite service should be achieved [3, 11]. As the metrics are the best method to assess and evaluate the quality of software, the metrics are needed for measuring the quality of composite services in SOD. Therefore, this section is proposing a set of metrics for estimating the quality of composite services design in order to aid in early detection of design flaws. The key factors in these quality measurements are the structure of SOD properties namely; abstraction, autonomy, cohesion, composability, contract, coupling, discoverability, reusability and statelessness [1, 15, 16].

This research work will propose quality measurement to evaluate the design of service oriented principle, which can affect the quality of service oriented design when it is designed improperly consequently, the reset development phases affected. The software quality metrics can be either basic metrics or derived metrics [55]. A basic metric is a simple metric defined as a function uses a single attribute for measuring the quality of software. While a derived metric is a complicated metric which defined as a function uses two or more basic metrics to quantify the quality of software.

3.1 Basic Metrics

The software quality metrics can be either basic metrics or derived metrics [55]. A basic metric is a simple metric defined as a function uses a single attribute for measuring the quality of software used as a first step to propose the derived metrics. There are many basic metrics in previous development paradigm such as OOC and CBC and this section can extend the basic metrics from previous paradigm with the characteristic of service-oriented paradigm. The metrics for previous development approaches such as OO and CB cannot be blindly applied to SOD without modification due to the special characteristics of service orientation [24]. However, the metrics presented in this paper are designed based on composite service design modeling (ComSDM) method [56], in order to solve the limitations in the existing metrics. Following sections provide the definition of each basic metrics.

1. Number of services (NS):

The number of services (NS) metric is a simple metric used to count the number of services in service-oriented system [57]. NS is the first indicator of system size, which can determine the complexity of

the system. The formal definition of this metric is provided in Equation 1.

$$NS(SOS) = \sum_{s \in SOS}^{n} 1 \quad \text{(Eq.1)}$$

This equation means, for each service (s) belong to the service-oriented system (SOS) increases the number of services (NS) by one. This metric is a simple measure used later to calculate the value of design size which can be used as the first indicator to estimate the system complexity. This metric is customized from number of classes metric in OOC and number of services metric in service-oriented system [57]. This metric is extended in order to cover the characteristics of service-oriented system in next metric due to missing of these characteristics in OOC. Also, the extensions cover the characteristics of the composite service because the previous metrics for measuring the quality of service design consider the atomic service rather than composite service. The complexity of system depends on the number of the services in this system, when the number of services increases the complexity also gets high. However, the NS is not only metric used to calculate the complexity of service-oriented system, but also the interaction between the services will affect the complexity of services-oriented design. Based on the service-oriented design the service can be either atomic or composite service. Usually, the composite service contains other basic services but, it's counted as one service in the service-oriented system. The atomic service for each composite service is considered as internal components of a composite service which calculate by the metric presented in Equation 2.

$$NS(ComS) = \sum_{s \in ComS}^{n} 1 \quad \text{(Eq. 2)}$$

Equation 2 means, for each service (s) belong to the composite service (ComS) in service-oriented system (SOS) increases the number of services (NS) by one. This metric counts the number of service only in composite service and used to calculate the internal interactions in composite service which determine the complexity of this service.

2. Number of operations (NO):

The number of operations (NO) is the second simple metric which counts all the number of operations as the other indicator for service-oriented system complexity. The NO in service considered very important, because its determine the granularity of services and complexity of system [15]. The service contains more operations indicates that this service is coarse granularity, whilst the service considered fine granularity when it contains fewer operations. The NO metric for counting the operations in each service is given in Equation 3.

$$NO(s) = \sum_{o \in s}^{n} 1 \quad \text{(Eq. 3)}$$

NO(s) is a set of all operations in a service (s). This metric counts all the operations in specified service (s). This Equation means, for each operation (o) belong to the service (s) increases the number of operations (NO) in this service by one. This metric is customized from number of operations metric in service-oriented system [15]. This metric is extended in order to cover the characteristics of composite service in the next metric due to missing of these characteristics in previous metrics for measuring the quality of service design. Further, to calculate the overall NO in service-oriented system a new metric is given in Equation 4.

$$NO(SOS) = \sum_{o \in SOS}^{n} 1 \quad \text{(Eq. 4)}$$

This metric counts all operations in the system from all the service. This equation means, for each service (s) belong to the service-oriented system (SOS) increase the number of operation in the service-oriented system (NO (SOS)) by adding together the number of operations (NO(s)) in each service. This metric is used to calculate the cohesion between the components of service-oriented system.

3. Provider (P):

The provider (P) is the service or operation that provides functionality for other services or operations. This metric is counting all the services or operation which proposes functionalities and used by other services or operations in a given services-oriented system.

$$P = \{(s,o) \in P | (s \in S) \wedge (o \in O) \wedge (s \wedge o) \neq \emptyset \wedge (s,o) \in R \wedge R\ is\ In\} \quad \text{(Eq. 5)}$$

provider in service-oriented system which can be A service (s) or an operation (o) is a provider (P) if and only if (s) or (o) is provided functionalities and these functionalities are used by other services or operations. This metric is for counting the entire used later in coupling and cohesion metrics.

4. Consumer (C):

The consumer (C) is the service or operation which is using the functionality that provided by other operations or services. This metric is counting all the services and operations which used or invoked the functionalities of other services or operations to achieve their tasks.

$$C(p) = \{(s,o) \in C | (s \in S) \wedge (o \in O) \wedge (p \in P) \wedge ((s \cup o) \wedge p) \in R \wedge R\ is\ Out\} \quad \text{(Eq. 6)}$$

A service (s) or an operation (o) is a consumer (C) which consumes the functionalities provided by the provider (p): if and only if (s) or (o) is consuming the

functionalities provided by other services or operations. This metric is for counting the entire consumer in service-oriented system which can be used later in coupling and cohesion metrics.

3.2 Weighting the Components of SOD

The service-oriented system contains many components among them services, operations and relationships. The service is the main component of service-oriented system and its structure is more complicated comparing with operations structure. Since the service can be atomic or composite services and contains many components such as operations and basic service. Some service-oriented system components are more importance than other due to their structure and interactions with other system components. Therefore, the weight method is used to distinguish between different components in service-oriented system by assigning different scale for components based on their importance. ComSDM method defined two types of interactions between service-oriented design components which are internal and external interaction.

Firstly, interaction between internal components of service which are interaction between two operations in the same services (IOR), the interaction between service and operation in the same service (IOSR) and interaction between two services in the same service (ISR). Secondly, interaction between external components of services, which cover the interaction between two operations belong to different services (EOR), interaction between operation and service belong different services (EOSR) and interaction between tow services belong to different services (ESR). The degree of interactions among services is higher than degree of interaction among operations. The service invocation leads to interact with more than one element within service, whereas calling operation only single unit communicate with other system components. The interactions among different services in service-oriented system are stronger than the interactions among operations with services as well as interactions among operations themselves. Thus, the interaction among services should be weighted higher, then the interaction between services and operations weighted medium and lastly the interactions among operations should be weighted lower as shown Table 3.

5. Importance of provider (IP):

Importance of provider (IP) is metric used to give weight for the operations and services. This metric is counting all the consumers, which depend on the provider by invoking its functionalities.

$$IP(p) = \sum_{i=1}^{c} C(p) * \text{weight value} \qquad \text{(Eq. 7)}$$

This metric is very significant and can used as a weight factor in the services and operations in service-oriented system. This metric gives the importance of the services and operations, according to the calls from other services or operations. The high value of IP (p) means the provider p is very important because many consumers use its functionalities. During the design of important provider the designers should take care because many services and operations are depending on it.

Table 3 Interaction weight

Internal Relationship	External relationship	Weight scale	Weight value
ISR	ESR	Higher	3
IOSR	EOSR	Medium	2
IOR	EOR	Lower	1

3.3 Derived Metrics

3.3.1 Coupling Metric

Coupling in service-oriented is defined as the interaction and dependency between the services in service-oriented systems. Coupling metrics measure the interaction dependency between the services and operations and it are calculated just by counting all direct relationships between services and its operations in service-oriented system and there is no consideration of indirect relationships (e.g. A service s1 calls service s2 but service s2 also call another service s3, in this case s1 calls s2 directly and calls s3 indirectly).

I. Direct coupling (DC):

The direct coupling is the direct interactions between the providers and consumers in a service-oriented system and calculated by counting the entire direct consumer for specific provider as appear in Equation 8.

$$DC(p) = C(p) \qquad \text{(Eq. 8)}$$

This means for each provider (p) counts all the direct call from all consumers (C).

II. Indirect coupling (IC):

The indirect coupling is the interactions between providers with direct and indirect consumer in service-oriented design. This metric is calculated by counting the direct consumers and then assume that all the consumers are providers and calculates their coupling.

$$IC(p) = DC(p) + \sum_{c(p) \in P} IC(c(p)) \qquad \text{(Eq. 9)}$$

Equation 8 shows the indirect coupling metric. This metric gives a better result than direct coupling because it takes into account both direct and indirect relationships and the result of coupling is more accurate than the previous metrics.

III. Coupling factor (CoupF):

The indirect coupling gives the result as a number and this number could not interpret by its self because this number may be gives a good indicator

if the system is big or worse if the system is small. However, new metric provided in Equation 9 compare the result of (IC) with service-oriented system size is needed to understand the value of this metric. Let f = NS (SOS) + NO (SOS) then:

$$CopF(p) = \frac{IC(p)}{f^2 - f} \quad \text{(Eq. 10)}$$

The principle of service-oriented system the design should be loosed coupling and the coupling will affect the complexity of the system. The results of these metrics give the designer indicator to the design coupling and how can improve the design to avoid the complexities. These metrics consider both direct and indirect interaction between the service-oriented system elements.

3.3.2 Cohesion Metrics

For any service-oriented system, cohesion estimates the degree to which the components of the system belong together and the strength of the relationships between operations in a service. In other word, cohesion estimates the difficulty of understanding the relationships between services and service operations.

a) Cohesion metric (CM):

This metric is measuring the degree of cohesion for specific service (s) in service-oriented system design. The border of this metric is only the service (s).

$$CM(s) = \{c(p) | (c \in C) \wedge (p \in P) \wedge (c \wedge p) \in s\} \quad \text{(Eq. 11)}$$

This means for each service (s) belong to service-oriented system counts all the consumers c (p) and consume the functionalities provided by providers (p). Which consumers (c) or providers (p) are belonging to this service (s).

b) Cohesion factor (CohF):

The cohesion metric gives the result as a number and this number could not interpreted by its self because this number may be gives good indicator if the system is big or bad if the system is small. However, new metric compare the result of (CM) with service-oriented system size is needed to understand the value of this metric. This metric is provided in Equation 11. Let f = NS(s) + NO(s) then

$$CohF(s) = \frac{CM(s)}{f^2 - f} \quad \text{(Eq. 12)}$$

The principle of service-oriented system the design should be tied cohesion and the cohesion will affect the complexity of the system. The results of these metrics give the designer indicator to the design cohesion and how can improve the design to avoid the complexities. These metrics consider the service to build the other services.

3.3.3 Complexity Metrics

Complexity measures the difficulty of understanding the interaction and relationships between the services and services operations.

1) Total complexity metric for a service:

For any service-oriented system, coupling and cohesion used to estimate the degree to which the components of the system belong together and the strength of the relationships between operations in a service. This metric calculates the complexity for a specific service (s).

$$TCM(s) = \frac{IC(s) + NS(s) + NO(s)}{CM(s)} \quad \text{(Eq. 13)}$$

2) Complexity factor (ComF):

This metric calculates the complexity factor from coupling and cohesion factor to give a better understanding of the complexity metric for a system.

$$ComF(s) = \frac{CopF(s)}{CohF(s)} \quad \text{(Eq. 14)}$$

3) Total complexity metric in a system:

This metric calculates the complexity of the entire system.

$$TCM(SOS) = \sum_{s \in SOS} TCM(s) * ComF(s) \quad \text{(Eq. 15)}$$

This means for each service (s) in service-oriented system add the result of total complexity metric (TCM) multiple complexity factor (ComF(s)).

3.3.4 Reusability Metric

The services in service-oriented system should be designed in a way through which the reusability of the system is increased. The reusability in service-oriented system is affected by two factors which are the direct consumers for the service and the degree of cohesiveness of the operations in the service [15]. The services have less direct interactions with other service components and higher cohesiveness between its operations are more reusable. Therefore, the direct coupling metric is used to measure the reusability of system by calculating the direct consumers of each service. The Equation 16 shows the calculation of Reusability Metric (RM) for each provider (p).

$$DC(p) = C(p) \quad \text{(Eq. 16)}$$

1. Reusability Factor

The reusability factor used to compare the reusability values with the system size by measuring the cohesion of operations. The Equation 17 represents the Reusability Factor (ResF) metric:

$$ResF = \frac{CM(SOS)}{DC(SOS)} \quad \text{(Eq. 17)}$$

4.0 VALIDATION

The theoretical validation of the software metrics normally uses the property-based approaches to ensure that the attributes supposed to be calculated by metrics is measured [55]. In this work the property-based software engineering measurement framework proposed by Briand et. al. [28, 58] is used. This approach was chosen for validation the proposed measurement theoretically in this thesis for some reasons among them, it is based on measurement theory and applied successfully by other researchers [59, 60]. In addition, it is comprehensive framework which defines the structural properties of software system mathematically which matches with the methodology of the proposed metrics in this thesis. However, to validate the proposed metrics theoretically using property-based approach there are six properties each metric has to satisfy these properties [28]. These properties include Nonnegative, Normalization, Null Value, Monotonicity, Marging of Services and Disjoint Service Additivity. Following subsection provides the result of these properties for the proposed metrics to demonstrate its satisfaction.

4.1 Theoretical Validation Results

The NS(SOS) and NS(ComS) metrics properties are verified theoretically using the properties-based approach. The result of these metrics is nonnegative, because the system in SOD and composite service design either has a set of service then the services counted and be positive or there is no service in the system which means the NS is zero never can be negative (Nonnegative). Similarly, the NO(SOS), NO(S), providers and consumers metrics never can be negative (Nonnegative). Likewise, for the derived metrics the DC, IC, CoupF, CM, and CohF metrics never can be negative (Nonnegative). In the same way, the complexity and reusability metrics never can be negative (Nonnegative).

4.1.1 Coupling Metrics Validation

PROPERTY COUPLING.1: Nonnegativity [28]. The proposed metrics for measuring coupling in this thesis depend on the external interactions between the components of system. When the servics in service-oriented system system have external interactions then the coupling should be positive. The DC, IC and CoupF are satisfied this property since the values obtained by these metrics can be zero when there is no external interaction between the components of system. However, under all circumstances the results of these metrics never can be negative. The mathematical demonstration of nonecative of the coupling metrics is provided as follow:

- DC(p) = C(p).
- If C(p) = Ø then DC(p) is zero, or more when C(p) is greater than Ø.
- Consequently, DC(p) ≥ 0.
- Accordingly, IC(p) and CoupF ≥ 0.

However, the coupling metrics provided in this research work are nonnegative.

PROPERTY COUPLING.2: Null Value [28]. DC, ID, and CoupF are null if there are no consumers (c) for each of the providers (p) in service-oriented system, which means there are no external interactions between the services in service-oriented system. The mathematical demonstration of null value of the coupling metrics is provided as follow:

- $The\ external\ C(p) = \emptyset \Rightarrow DC, IC\ and\ CoupF = 0.$

The *Null Value* property is satisfied, since all the coupling metrics provide null value when there are no interactions between the system components.

PROPERTY COUPLING.3: Monotonicity [28]. The coupling values do not decreased by adding more external interactions or dependencies between services and operations in service-oriented system. This property is satisfied in coupling metrics which the coupling between system components is increased by adding new external interactions (*Monotonicity*).

PROPERTY COUPLING.4: Merging of Services [28]. The real name of this property is *Merging of Modules*, but changed in this research work for *Merging of Services* because the system in service-oriented has services instead of modules. The result of coupling metrics obtained by integrating two or more services in one service (*composite service*) is less than or equal the sum of coupling metrics result of the two or more original services since some dependencies between the services may have disappeared (*Merging of Services*). The proposed metrics satisfied this property of *Merging of Services.*

PROPERTY COUPLING.5: Disjoint Service Additivity [28]. The real name of this property is Disjoint Module Additivity, but changed in this research work to Disjoint Service Additivity for same reason above. The result of coupling metrics obtained by *composing two or more discrete* services in composite service is equal to the sum of coupling metrics result of the two or more original services which are composed together (*Disjoint Service Additivity*). In other words, composing unrelated services which have not interactions among themselves in a composite service will not decrees the overall coupling of composite service because the composition is not reduced the external interactions that can affect the coupling of service. The proposed metrics satisfied this property of *Disjoint Service Additivity,* because the metrics results show that the coupling is reduced only when the related services are composed together.

PROPERTY COUPLING.6: Normalization. Normalization means the values of metrics are normalized between 0 and 1 in order to provide meaningful comparisons which can facilitate the understanding the values of

the metrics, since they belong to the same system. The normalization property is proposed to validate the cohesion in [28], but in this section used to validate the coupling because coupling the results of coupling and cohesion metrics are same. This property is satisfied since CoupF metric is normalized between 0 and 1 *(Normalization*).

4.1.2 Cohesion Metrics Validation

PROPERTY COHESION.1: Nonnegativity and Normalization [28]. The proposed metrics for measuring cohesion in this thesis depend on the internal interactions between the components of services in the system. When the service has internal interaction then the cohesion should be positive. The CM and CohF are satisfied this property since the values obtained by these metrics can be zero when there is no internal interaction between the components of system. However, under all circumstances the results of these metrics never can be negative *(Nonnegative*). The metric CohF is normalized between 0 and 1 *(Normalization*). The mathematical demonstration of nonnegative and normalization of the cohesion metrics is provided as follow:

- $CM(s) = \{c(p) | (c \in C) \wedge (p \in P) \wedge (c \wedge p) \in s\}$
- If *C(p)* = Ø then *CM(s)* is zero, or more when *C(p)* is greater than Ø. *C(p)* is the internal interactions between the consumer (c) and provider (p), which they are belong to service (s).
- Consequently, *CM(s)* ≥ *0*.
- Accordingly, *CohF* ≥ *0*.
- The *CohF* is normalized by divided the result of cohesion metric on the maximum expected cohesion.

The *Nonnegative and Normalization* property is satisfied, since all the cohesion metrics provide nonnegative value and the CohF metric is normalized between 0 and 1.

PROPERTY COHESION.2: Null Value [28]. CM and CohF are null if there are no consumers (c) for each of the providers (p) in service (s), which means there are no internal interactions between the components belonging to the same service. The mathematical demonstration of null value of the cohesion metrics is provided as follow:

$$The\ internal\ \mathrm{C(p)} = \emptyset \Rightarrow \mathrm{CM\ and\ CohF} = 0.$$

The *Null Value* property is satisfied, since all the cohesion metrics provide null value when there are no interactions between the system components.

PROPERTY COHESION.3: Monotonicity [28]. The cohesion metrics values do not decreased by adding more internal interactions or dependencies between the components belonging to the same service in service-oriented system. This property is satisfied in cohesion metrics which the cohesion between service components is increased by adding new internal interactions (*Monotonicity*). For example, suppose that S1 (si, S_{s1}, O_{s1}, R_{s1}) and S2 (si, S_{s2}, O_{s2}, R_{s2}) are two composite services and the cohesion of S1 is equal to the cohesion of S2. When adding new internal interaction between the components of the service S1 then [Cohesion(S1) ≥ Cohesion(S2)].

PROPERTY COHESION.4: Cohesive Service [28]. The real name of this property is Cohesive Module, but altered in this research work to Cohesive Service for same reason above. The result of cohesion metrics obtained by *composing two or more discrete* services in composite service is not greater than the sum of cohesion metrics results of the two or more original services which are composed together (*Disjoint Service Additivity*). In other words, composing unrelated services which have not interactions among themselves in a composite service will not increase the overall cohesion of the composite service because the composition is not increase the internal interactions but increase the number of internal components of the service which negatively affected the cohesion of service. The proposed metrics satisfied this property of *Cohesive Service,* because the metrics results show that the cohesion is increased only when the related services are composed together.

4.1.3 Complexity Metrics Validation

PROPERTY COMPLEXITY.1: Nonnegativity [28]. The proposed metrics for measuring complexity in this thesis depend on the interactions between the components of the system. When the system has interactions between its components then the complexity should be positive. The complexity metric for a service, ComF and complexity metric in a system are satisfied this property since the values obtained by these metrics can be zero when there is no interaction between the components of the system. However, under all circumstances the results of these metrics never can be negative. The mathematical demonstration of nonnegative of the complexity metrics is provided as follow:

- If IC(s) and CM(s) = Ø then TCM(s) is zero because there is no interactions in this system, or more when the interaction on the system is greater than Ø.
- Accordingly, TCM(SOS) and ComF ≥ 0.

However, the complexity metrics provided in this thesis are nonnegative.

PROPERTY COMPLEXITY.2: Null Value [28]. The total complexity metric for a service ComF and total complexity metric in a system are null if there are no consumers (c) for each of the providers (p), which means there are no interactions between the components of the system. The mathematical demonstration of null value of the complexity metrics is provided as follow:

$$\mathrm{C(p)} = \emptyset \Rightarrow \mathrm{TCM(s), ComF(s) and\ TCM(SOS)} = 0.$$

The *Null Value* property is satisfied, since all the complexity metrics provide null value when there are no interactions between the system components.

PROPERTY COUPLING.3: Monotonicity [28]. The complexity metrics values do not decreased by adding more interactions or dependencies between services and operations in service-oriented system. This property is satisfied in complexity metrics which the coupling between system components is increased by adding new interactions (*Monotonicity*).

PROPERTY COMPLEXITY.4: Disjoint Service Additivity [28]. The real name of this property is *Disjoint Module Additivity*, but changed in this research work to *Disjoint Service Additivity* for same reason above. The result of complexity metrics obtained by *composing two or more discrete* services in composite service is equal to the sum of complexity metrics results of the two or more original services which are composed together (*Disjoint Service Additivity*). In other words, composing unrelated services which have not interactions among themselves in a composite service will not decrese the overall complexity of composite service because the composition is not reduced the interactions that can affect the complexity of service. The proposed metrics satisfied this property of *Disjoint Service Additivity,* because the metrics results show that the complexity is reduced only when the related services are composed together.

PROPERTY COMPLEXITY.5: Merging of Services [28]. The real name of this property is *Merging of Modules*, but altered in this research work for *Merging of Services* because the system in service-oriented has services instead of modules. The results of complexity metrics obtained by integrating two or more services in one service (*composite service*) is less than or equal the sum of complexity metrics results of the two or more original services since some dependencies between the services may have disappeared (*Merging of Services*). The proposed metrics satisfied this property of *Merging of Services.*

4.1.4 Reusability Metrics Validation

PROPERTY REUSABILITY.1: Nonnegativity. The reusability metrics are validated theoretically using the properties proposed for validating the complexity metrics because there no proposed properties for reusability in [28]. The proposed metrics for measuring reusability in this thesis depend on the direct external interactions between the components of system with specific service (s) and the internal interactions between its components. When the system has interaction then the reusability should be positive. The RM and ReuF are satisfied this property since the values obtained by these metrics can be zero when there is no interaction between the components of system. However, under all circumstances the results of these metrics never can be negative. The mathematical demonstration of nonnegative of the coupling metrics is provided as follow:

- RM(s) = C(s).
- If C(s) = Ø then RM(s) is zero, or more when C(s) is greater than Ø.
- Consequently, ReuF(SOS) ≥ 0.

However, the reusability metrics provided in this thesis are nonnegative.

PROPERTY REUSABILITY.2: Null Value. RM(s) is null if there are no consumers (c) for each of the service (s) which means there are no external interactions between the service(s) and the other components in service-oriented system. Further, the ReuF(SOS) is null if the reusability of all the services in system is null. The mathematical demonstration of null value of the coupling metrics is provided as follow:
$\forall\, Service(s) \in SOS, \mathrm{C(s)} = \emptyset \Rightarrow \mathrm{RM(s)}$ and $\mathrm{ReuF(SOS)} = 0$.
The *Null Value* property is satisfied, since all the reusability metrics provide null value when there are no interactions between the system components.

PROPERTY REUSABILITY.3: Monotonicity [28]. The reusability metrics values do not decreased by adding more external interactions or dependencies for service(s) and adding more internal interactions between the components of service(s). This property is satisfied in reusability metrics which the reusability of services is increased by adding new external interactions for the service (*Monotonicity*).

PROPERTY REUSABILITY.4: Merging of Services [28]. The result of reusability metrics obtained by integrating two or more services in one service (*composite service*) is greater than or equals the sum of reusability metrics results of the two or more original services since some dependencies between the services may have disappeared (*Merging of Services*). The proposed metrics for reusability are satisfied this property of *Merging of Services.*

PROPERTY REUSABILITY.5: Disjoint Service Additivity [28]. The result of reusability metrics obtained by *composing two or more discrete* services in composite service is less than the sum of reusability metrics results of the two or more original services which are composed together (*Disjoint Service Additivity*). In other words, composing unrelated services which have not interactions among themselves in a composite service will not increases the overall reusability of composite service because the composition is not reduced the external interactions that can affect the reusability of service. The proposed metrics satisfied this property of *Disjoint Service Additivity,* because the metrics results show that the reusability is increased only when the related services are composed together.

5.0 DISCUSSION

Software quality measurement is a necessary target of software engineering and, in addition, has considerable importance in the context of SOD since it determines how the quality requirements for composite service should be achieved [3, 11]. As the metrics are the best method to assess and evaluate the quality of software, the metrics are needed for measuring the quality of composite services in SOD. The SOD is the key phase, and assessing the quality at this level is very essential to reduce the cost and effort of the implementation phase and enhance the quality of software applications.

In order to assess the quality of SOD many metrics are proposed in the literature but these metrics are constructed based on the characteristic of the previous development approaches such as OOC and CBC [23]. Similarly, the existing metrics for the service-oriented design are still at a preliminary stage [12]. The metrics development for approaches such as OOC and CBC, do not work well. These metrics are also not good for service-oriented systems without modification due to unique characteristics of service orientation [7]. Therefore, this chapter is proposing a set of metrics for estimating the quality of composite services design in order to aid in early detection of design flaws. The proposed metrics take into account the characteristics of the previous development approaches as well as the composite service in service-oriented system. The key factors in these quality measurements are the structure of SOD properties namely; abstraction, autonomy, cohesion, composability, contract, coupling, discoverability, reusability and statelessness [1, 15, 16].

Further, the previous metrics do not consider the service that is built from other services (composite services) and only consider the operations as building blocks for service-oriented system [23]. Furthermore, these metrics are not able to measure the quality of composite service design. The proposed metrics consider the atomic services and composite services as building blocks for the design of service-oriented system.

In addition, the previous do not consider the indirect relationships between service-oriented elements to measure the quality of composite service and only the direct interactions are considered [25]. The indirect relationships are very important to give more accurate results in measuring the coupling between composite services in service-oriented system. The proposed metrics produce new equations to consider the indirect interactions between the elements of service-oriented system as well as the direct relationships.

In most cases, metrics were used to calculate the quality attributes of SOD, such as coupling and cohesion but were unable to establish relationships between the attributes [19, 24]. The proposed metrics have been succeeded to establish relationships between the attributes of service-oriented design to measure the reusability and complexity of the system from coupling and cohesion attributes.

6.0 CONCLUSION

Service-Oriented has been applied successfully in the development of many types of software. Therefore, many metrics are proposed to assess and evaluate the SOD in order to improve the quality of service-oriented systems. This studies was described the features of SOD and principles to facilitate the measurement of the quality of composite service design and defined the component of service-oriented system. This paper defines two types of software metrics which are basic and derived metrics. In this paper a set of basic metrics is proposed and used for proposing derived metrics to evaluate the coupling, cohesion, complexity and reusability of composite service design. The result of this study shows how these metrics calculate their values and why these metrics are important. These metrics add a new contribution to assess the quality of composite service design mainly for coupling, cohesion, complexity and reusability. These metrics are validated theoretically. The results show that these metrics are able to measure the quality of composite service design. Moreover, the proposed metrics can be used as a first step to propose a quality measurement model for composite service design by proposing design style selection method for composite service in the next stage.

Acknowledgement

We would like to thank Universiti Teknologi Malaysia for sponsoring the research through the RUG grant with vote number 05H83 and providing the facilities and support for the research.

References

[1] Erl, T. 2005. *Service-oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR.

[2] Kim, T., C. K. Chang, and S. Mitra. 2010. Design of Service-Oriented Systems Using SODA. *Services Computing, IEEE Transactions on.* 3(3): 236-249.

[3] Zhang, Q. Q. and X. K. Li. 2009. Complexity Metrics for Service-Oriented Systems. 2009 *Second International Symposium on Knowledge Acquisition and Modeling: Kam.* 3: 375-378.

[4] Mohamad, R., *et al.* 2012. Service Identification Guideline for Developing Distributed Embedded Real-time Systems. *Software, IET.* 6(1): 74-82.

[5] Aziz, M., *et al*. 2013. Service Based Meta-model for the Development of Distributed Embedded Real-time Systems. *Real-Time Systems*. 49(5): 563-579.

[6] McHeick, H. and Q. Yan. 2012.Quality Attributes and Design Decisions in Service-oriented Computing. In *Innovations in Information Technology (IIT), 2012 International Conference on*.

[7] Perepletchikov, M. and C. Ryan. 2011. A Controlled Experiment for Evaluating the Impact of Coupling on the Maintainability of Service-Oriented Software. *Software Engineering, IEEE Transactions on*. 37(4): 449-465.

[8] Daghaghzadeh, M., A. B. Dastjerdi, and H. Daghaghzadeh. 2011. A Metric for Measuring Degree of Service Cohesion in Service Oriented Designs. *International Journal of Computer Science*. 8.

[9] Elhag, A. A. M., *et al.* 2013. Problems and Future Trends of Software Process Improvement in Some Sudanese Software Organizations. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*.

[10] Goeb, A. and K. Lochmann. 2011. *A Software Quality Model for SOA*. ACM.

[11] Nematzadeh, H., *et al.* 2014. QoS Measurement of Workflow-Based Web Service Compositions Using Colored Petri Net. *The Scientific World Journal.* 2014: 14.

[12] Gebhart, M. and S. Abeck. 2011. Quality-oriented Design of Services. *International Journal on Advances in Software.* 4(1 and 2): 144-157.

[13] Feuerlicht, G. 2011. *Simple Metric for Assessing Quality of Service Design Service-Oriented Computing*, E. Maximilien, et al., Editors. Springer Berlin / Heidelberg. 133-143.

[14] Kruger, I. H. and R. Mathew. 2004. Systematic Development and Exploration of Service-oriented Software Architectures. In Software Architecture, 2004. WICSA 2004. *Proceedings. Fourth Working IEEE/IFIP Conference on*.

[15] Sindhgatta, R., B. Sengupta, and K. Ponnalagu. 2009. *Measuring the Quality of Service Oriented Design Service-Oriented Computing*. L. Baresi, C.-H. Chi, and J. Suzuki, Editors. Springer Berlin / Heidelberg. 485-499.

[16] Reddy, V., *et al.* 2009. Evaluating Legacy Assets in the Context of Migration to SOA. *Software Quality Journal.* 17(1): 51-63.

[17] Rostampour, A., *et al.* 2011. Measures of Structural Complexity and Service Autonomy. *IEEE.*

[18] Rostampour, A., *et al.* 2010. A Metric for Measuring the Degree of Entity-centric Service Cohesion. In Service-Oriented Computing And Applications (SOCA). *2010 IEEE International Conference on*.

[19] Perepletchikov, M., C. Ryan, and K. Frampton. 2007. *Cohesion Metrics for Predicting Maintainability of Service-Oriented Software*.

[20] Kazemi, A., *et al.* 2011. An Information Retrieval Based Approach for Measuring Service Conceptual Cohesion. in Quality Software (QSIC). *2011 11th International Conference on*.

[21] Alahmari, S., E. Zaluska, and D.C. De Roure. 2011. A Metrics Framework for Evaluating SOA Service Granularity. in Services Computing (SCC). *2011 IEEE International Conference on*.

[22] Si Won, C. and K. Soo Dong. 2008. A Quality Model for Evaluating Reusability of Services in SOA. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*.

[23] Perepletchikov, M., *et al.* 2007. A Formal Model of Service-Oriented Design Structure. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*.

[24] Perepletchikov, M., *et al.* 2007. Coupling Metrics for Predicting Maintainability in Service-Oriented Designs. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*.

[25] Qian, M., *et al.* 2009. Evaluating Service Identification with Design Metrics on Business Process Decomposition. In *Services Computing, 2009. SCC '09. IEEE International Conference on*.

[26] Chidamber, S. R., D. P. Darcy, and C. F. Kemerer. 1998. Managerial Use of Metrics for Object-oriented Software: An Exploratory Analysis. Software Engineering. *IEEE Transactions on.* 24(8): 629-639.

[27] Fenton, N. E. and M. Neil. 2000. Software Metrics: Roadmap. In *Proceedings of the Conference on The Future of Software Engineering*ACM: Limerick, Ireland. 357-370.

[28] Briand, L. C., S. Morasca, and V. R. Basil. 1996. Property-Based Software Engineering Measurement. *Software Engineering, IEEE Transactions on*. 22(1): 68-86.

[29] Erl, T., Soa. 2008. *Principles of Service Design*. 1. Prentice Hall.

[30] Hirzalla, M., J. Cleland-Huang, and A. Arsanjani. 2009. A Metrics Suite for Evaluating Flexibility and Complexity in Service Oriented Architectures. *Service-Oriented Computing - Icsoc 2008 Workshops.* 5472: 41-52.

[31] Krafzig, D., K. Banke, and D. Slama. 2005. *Enterprise SOA: Service-oriented Architecture Best Practices*. Prentice Hall PTR.

[32] Perepletchikov, M., C. Ryan, and Z. Tari. 2010. The Impact of Service Cohesion on the Analyzability of Service-Oriented Software. Services Computing. *IEEE Transactions on.* 3(2): 89-103.

[33] Bingu, S., *et al.* 2008. A Design Quality Model for Service-Oriented Architecture. In *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*.

[34] Moayerzadeh, A. and E. Yu. 2009. *A Goal-oriented Representation of Service-oriented Software Design Principles*.

[35] Ma, Y.-F., H.X. Li, and P. Sun. 2007. A Lightweight Agent Fabric for Service Autonomy. In *Service-Oriented Computing: Agents, Semantics, and Engineering*. Springer. 63-77.

[36] Perepletchikov, M., C. Ryan, and K. Frampton. 2006. Towards the Definition and Validation of Coupling Metrics for Predicting Maintainability in Service-Oriented Designs on the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. R. Meersman, Z. Tari, and P. Herrero, Editors. Springer Berlin/Heidelberg. 34-35.

[37] Kazemi, A., *et al.* 2011. Measuring the Conceptual Coupling of Services Using Latent Semantic Indexing. in Services Computing (SCC). *2011 IEEE International Conference on*.

[38] Bansiya, J. and C. G. Davis. 2002. A Hierarchical Model for Object-oriented Design Quality Assessment. *Software Engineering, IEEE Transactions on*. 28(1): 4-17.

[39] Daghaghzadeh, M., A. B. Dastjerdi, and H. Daghaghzadeh. A Metric for Measuring Degree of Service Cohesion in Service Oriented Designs. *International Journal of Computer Science*. 8.

[40] Si Won, C., H. Jin Sun, and K. Soo Dong. 2007. QoS Metrics for Evaluating Services from the Perspective of Service Providers. in e-Business Engineering, 2007. *ICEBE 2007. IEEE International Conference on*.

[41] Perepletchikov, M., C. Ryan, and K. Frampton. 2007. Cohesion Metrics for Predicting Maintainability of Service-Oriented Software. In *Quality Software, 2007. QSIC '07. Seventh International Conference on*.

[42] Rossi, P. and Z. Tari. 2007. Software Metrics for the Efficient Execution of Mobile Services. In *Emerging Web Services Technology*. Springer. 135-152.

[43] Dimitoglou, M. J. M. G. 2008. *A Service Oriented Architecture Complexity Metric, Based on Statistical Hypothesis Testing.*

[44] Hofmeister, H. and G. Wirtz. 2008. Supporting Service-Oriented Design with Metrics. In *Enterprise Distributed Object Computing Conference, 2008. EDOC '08. 12th International IEEE*.

[45] Bakhshi, M., F. Mardukhi, and N. Nematbakhsh. 2010. A Fuzzy-based Approach for Selecting the Optimal Composition of Services According to User Preferences. In *Computer and Automation Engineering (ICCAE). 2010 The 2nd International Conference on*.

[46] Gebhart, M. and S. Abeck. 2011. Metrics for Evaluating Service Designs Based on Soam. *International Journal on Advances in Software.* 4(1 and 2): 61-75.

[47] Gebhart, M., *et al.* 2010. Evaluation of Service Designs Based on SoaML. In *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*.
[48] Meeyeon, L., *et al.* 2010. A Design-Phase Quality Model of U-Service Ontology for Evaluating Dynamic Service Composition. In *Multimedia and Ubiquitous Engineering (MUE), 2010 4th International Conference on*.
[49] Feuerlicht, G. 2011. Simple Metric for Assessing Quality of Service Design. In *Service-Oriented Computing*. Springer. 133-143.
[50] Kazemi, A., *et al.* 2011. A Metric Suite for Measuring Service Modularity. In *Computer Science and Software Engineering (CSSE), 2011 CSI International Symposium on*.
[51] Kai, Q., L. Jigang, and F. Tsui. 2006. Decoupling Metrics for Services Composition. In *Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on*.
[52] Karhikeyan, T. and J. Geetha. 2012. A Metrics Suite and Fuzzy Model for Measuring Coupling in Service Oriented Architecture. In *Recent Advances in Computing and Software Systems (RACSS), 2012 International Conference on*.
[53] Khoshkbarforoushha, A., *et al.* 2010. Towards a Metrics Suite for Measuring Composite Service Granularity Level Appropriateness. In *Services (SERVICES-1), 2010 6th World Congress on*.
[54] Mohammed Elhag, A. A. and R. Mohamad. 2014. Metrics for Evaluating the Quality of Service-oriented Design. In *Software Engineering Conference (MySEC), 2014 8th Malaysian*.
[55] Montagud, S., S. Abrahão, and E. Insfran. 2012. A Systematic Review of Quality Attributes and Measures for Software Product Lines. *Software Quality Journal.* 20(3-4): 425-486.
[56] Elhag, A. A. M., *et al.* 2015. *A Systematic Composite Service Design Modeling Method Using Graph-Based Theory.*
[57] Gao, H., *et al.* 2009. Service-Oriented Modeling Method for the Development of an E-Commerce Platform. In *E-Business and Information System Security, 2009. EBISS'09. International Conference on*. IEEE.
[58] Briand, L.C., *et al*. 1998. A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems. In Software Metrics Symposium, Metrics 1998. *Proceedings. Fifth International*.
[59] Rossi, P. and G. Fernandez. 2003. Definition and Validation of Design Metrics for Distributed Applications. In *Software Metrics Symposium, Proceedings. Ninth International*.
[60] Morasca, S. 1999. Measuring Attributes of Concurrent Software Specifications in Petri Nets. In *Software Metrics Symposium, 1999. Proceedings. Sixth International*.