FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Designing a business platform using microservices.

Rajendra Kharbuja

FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Designing a business platform using microservices.

Entwerfen einer Business-Plattform mit microservices.

Author:	Rajendra Kharbuja
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Manoj Mahabaleshwar
Submission Date:	

Acknowledgments

Abstract

The microservices architecture provides various advantages such as agility, independent scalability etc., as compared to the monolith architecture and thus has gained a lot of attention.However, implementing microservices architecture is still a challenge as many concepts within the microservices architecture including granularity and modeling process are not yet clearly defined and documented. This research attempts to provide a clear understanding of these concepts and finally create a comprehensive guidelines for implementing microservices.
Various keywords from definitions provided by different authors are taken and cateogorized into various conceptual areas. These concepts along with the keywords are researched thoroughly to understand the microservices architecture. Additionally, the three important drivers: quality attributes, constraints and principles, are also focussed for creating the guidelines.
The findings of this research indicate that eventhough microservices emphasize on the concept of creating small services, the notion of appropriate granularity is more important and depends upon four basic concepts which are : single responsibility, autonomy, infrastructure capability and business value. Additionally, the quality attributes such as coupling, cohesion etc should also be considered for identification of microservices. Furthermore, in order to identify microservices, either the domain driven design approach or the use case refactoring approach can be used. Both these approaches can be effective in identifying microservices, but the concept of bounded context in domain driven design approach identifies autonomous services with single responsibility. Apart from literature, a detail study of the architectural approach used in industry named SAP Hybris is conducted. The interviews conducted with their key personnels has given important insight into the process of modeling as well as operating microservices. Moreover, the challenges for implementing microservices as well as the approach to tackle them are done based on the literature review and the interviews done at SAP Hybris.
Finally, the findings are used to create a detail guidelines for implementing microservices. The guidelines captures how to approach modeling microservices architecture and how to tackle its operational complexities.

Contents

1 Granularity

In this chapter, a detailed concept regarding size of microservices will be discussed. A major focus would be to research various qualitative aspects of granularity of microservices. Section 1.2 lists various principles which can guide to model correct size considering various important aspects. Later, in Section 1.3, various interpretations defining dimensions of microservices are presented. Finally, the secion provides a complete picture of various factors which determine granularity.

1.1 Introduction

Granularity of a service is often ambiguous and has different interpretation. In simple terms, it refers to the size of the service. However, the size itself can be vague. It can neither be defined as a single quantitative value nor it can be defined in terms of single dependent criterion. It is difficult to define granularity in terms of number because the concepts defining granularity are subjective in nature. If we choose an activity supported by the service to determine its granularity then we cannot have one fixed value instead a hierarchical list of answers; where an activity can either refer to a simple state change, any action performed by an actor or a complete business process. [Linnp], [Hae+np]

Although, the interest upon the granularity of a component or service for the business users only depends upon their business value, there is no doubt that the granularity affects the architecture of a system. The honest granularity of a service should reflect upon both business perspective and should also consider the impact upon the overall architecture.

If we consider other units of software application, we come from object oriented to component based and then to service oriented development. Such a transistion has been considered with the increase in size of the individual unit. The increase in size is contributed by the interpretation or the choice of the abstraction used. For example, in case of object oriented paradigm, the abstraction is chosen to represent close impression of real world objects, each unit representing fine grained abstraction with some attributes and functionalities.

Such abstraction is a good approach towards development simplicity and understanding, however it is not sufficient when high order business goals have to be implemented.

It indicates the necessity of coarser-grained units than units of object oriented paradigm. Moreover, component based development introduced the concept of business components which target the business problems and are coarser grained. The services provide access to application where each application is composed of various component services. [Linnp]

1.2 Basic Principles to Service Granularity

In addition to quantitative perspective on granularity, it can be helpful to approach the granularity qualitatively. The points below are some basic principles derived from various scientific and research papers, which attempt to define the qualitative properties of granularity. Hopefully, these principles will be helpful to come with the service of right granularity.

1. The correct granularity of a component or a service is dependent upon the time. The various supporting technologies that evolve during time can also be an important factor to define the level of decomposition. For eg: with the improvement in virtualization, containerization as well as platform as a service technologies, it is fast and easy for deployment automation which supports multiple fine-grained services creation.[HS00]

2. A good candidate for a service should be independent of the implementation but should depend upon the understandability of domain experts.[Hae+np; HS00]

3. A service should be an autonomous reusable component and should support various cohesion such as functional (group similar functions), temporal(change in the service should not affect other services), run-time(allocate similar runtime environment for similar jobs; eg. provide same address space for jobs of similar computing intensity) and actor (a component should provide service to similar users). [Hae+np], [HS00]

4. A service should not support huge number of operations. If it happens, it will affect high number of customers on any change and there will be no unified view on the functionality. Furthermore, if the interface of the service is small, it will be easy to maintain and understand.[Hae+np], [RS07]

5. A service should provide transaction integrity and compensation. The activities supported by a service should be within the scope of one transaction. Additionally, the compensation should be provided when the transaction fails. If each operation provided by the service map to one transaction, then it will improve availability and fault-recovery. [Hae+np], [Foo05] [BKM07]

6. The notion of right granularity is more important than that of fine or coarse. It depends upon the usage condition and moreover is about balancing various qualities such as reusability, network usage, completeness of context etc. [Hae+np], [WV04]

7. The level of abstraction of the services should reflect the real world business activities. Doing so will help to map business requirements and technical capabilities. [RS07]

8. If there are ordering dependencies between operations, it will be easy to implement and test if the dependent operations are all combined into a single service. [BKM07]

9. There can be two better approaches for breaking down an abstraction. One way is to separate redundant data or data with different semanting meaning. The other approach is to divide services with limited control and scope. For example: A Customer Enrollment service which deals with registration of new customers and assignment of unique customer ID can be divided into two independent fine-grained services: Customer Registration and ID Generation, each service will have limited scope and separate context of Customer.[RS07]

10. If there are functionalities provided by a service which are more likely to change than other functionalities. It is better to separate the functionality into a fine-grained service so that any further change on the functionality will affect only limited number of consumers. [BKM07]

1.3 Dimensions of Granularity

As already mentioned in Section 1.1, it is not easy to define granularity of a service quantitatively. However, it can be made easier to visualize granularity if we can project it along various dimensions, where each dimension is a qualifying attribute responsible for illustrating size of a service. Eventhough the dimensions discussed in this section will not give the precise quantity to identify granularity, it will definitely give the hint to locate the service in granularity space. It will be possible to compare the granularity of two distinct services. Moreover, it will be interesting and beneficial to know how these dimensions relate to each other to define the size.

1.3.1 Dimension by Interface

One way to define granularity is by the perception of the service interface as made by its consumer. The various properties of a service interface responsible to define its size are listed below.

1. Functionality: It qualifies the amount of functionality offered by the service. The functionality can be either default functionality, which means some basic group of logic or operation provided in every case. Or the functionality can be parameterized and depending upon some values, it can be optionally provided. Depending upon the functionality volume, the service can be either fine-grained or course-grained than other service. Considering functionality criterion, a service offering basic CRUD functionality is fine-grained than a service which is offers some accumulated data using orchestration. [Hae+np]

2. Data: It refers to the amount of data handled or exchanged by the service. The data granularity can be of two types. The first one is input data granularity, which is the amount of data consumed or needed by the service in order to accomplish its tasks. And the other one is ouput data granularity, which is the amount of data returned by the service to its consumer. Depending upon the size and quantity of business object/objects consumed or returned by the service interface, it can be coarse or fine grained. Additionally, if the business object consumed is composed of other objects rather than primitive types, then it is coarser-grained. For example: the endpoint "PUT customers/C1234" is coarse-grained than the endpoint "PUT customers/C1234/Addresses/default" because of the size of data object expected by the service interface. [Hae+np]

3. Business Value: According to [RKKnp], each service is associated with an intention or business goal and follows some strategy to achieve that goal. The extent or magnitude of the intention can be perceived as a metric to define granularity. A service can be either atomic or aggregate of other services which depends upon the level of composition directly influenced by the extent of target business goal. An atomic service will have lower granularity than an aggregate in terms of business value. For example, sellProduct is coase-grained than acceptPayment, which is again coarse-grained than validateAccountNumber. [Hae+np]

1.3.2 Dimension by Interface Realization

The Section 1.3.1 provides the aspect of granularity with respect to the perception of customer to interface. However, there can be different opinion regarding the same properties, when it is viewed regarding the imlementation. This section takes the same aspects of granularity and try to analyze them when the services are implemented.

1. Functionality: In Section 1.3.1, it was mentioned that an orchestration service has higher granularity than its constituent services with regard to default functionality. If the realization effort is focused, it may only include compositional and/or compensation logic because the individual tasks are acomplshed by the constituent service interfaces. Thus, in terms of the effort in realizing the service it is fine-grained than from the view point of interface by consumer. [Hae+np]

2. Data: In some cases, services may utilize standard message format. For example: financial services may use SWIFT for exchanging finanicial messages. These messages are extensible and are coarse grained in itself. However, all the data accepted by the service along with the message may not be required and used in order to fulfil the business goal of the service. In that sense, the service is coarse-grained from the viewpoint of consumer but is fine-grained in realization point of view. [Hae+np]

3. Business Value: It can make a huge impact when analyzing business value of service if the realization of the service is not considered well. For example: if we consider data management interface which supports storage, retrieval and transaction of data, it can be considered as fine-grained because it will not directly impact the business goals. However, since other services are very dependent in its performance, it becomes necessary to analyse the complexity associated with the implementation, reliablity and change the infrastructure if needed. These comes with additional costs, which should well be analyzed. From the realization point of view, the data service is coarse-grained than its view by consumer. [Hae+np]

Principle: A high Functionality granularity does not necessarily mean high business value granularity
It may seem to have direct proportional relationship between functionality and business value granularity. The business value of a service reflects business goals however the functionality refers to the amount of work performed by the service. A service to show customer history can be considered to have high functionality granularity because of the involved time period and database query. However, it is of very low business value to the enterprise. [Hae+np]

1.3.3 R^3 Dimension

Keen [Kee91] and later Weill and Broadbent [WB98] introduced a separate group of criteria to measure granularity of service. The granularity was evaluated in terms of two dimensions as shown in the Figure 1.1

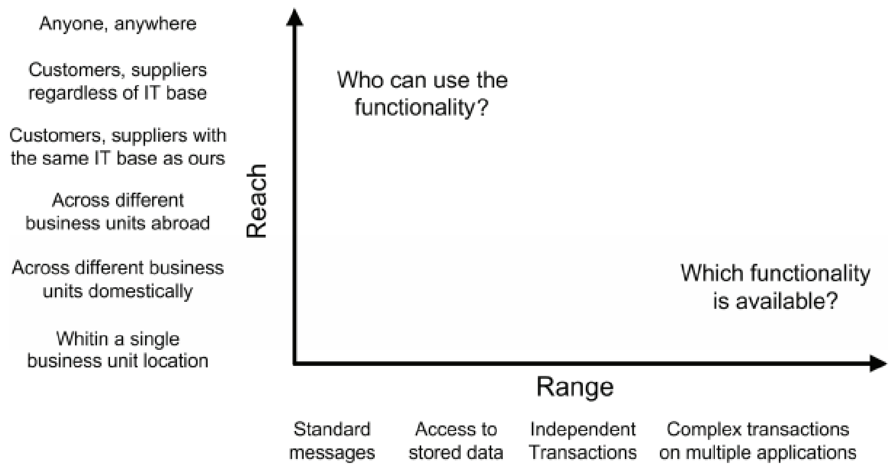
Anyone, anywhere
Customers, suppliers regardless of IT base
Customers, suppliers with the same IT base as ours
Across different business units abroad
Across different business units domestically
Whitin a single business unit location
Reach
Who can use the functionality?
Which functionality is available?
Range
Standard messages
Access to stored data
Independent Transactions
Complex transactions on multiple applications

Figure 1.1: Reach and Range model from [Kee91; WB98]

1. Reach: It provides various levels to answer the question "Who can use the functionality? ". It provides the extent of consumers, who can access the functionality provide by the service. It can be a customer, the customers within an organization, supplier etc. as given by the Figure 1.1.

2. Range: It gives answer to "Which functionality is available? ". It shows the extent to which the information can be accessed from or shared with the service. The levels of information accessed are analyzed depending upon the kind of business activities accessible from the service. It can be a simple data access, a transaction, message transfer etc as shown in the Figure 1.1 The 'Range' measures the amount of data exchanged in terms of the levels of business activities important for the organization. One example for such levels of activities with varying level of 'Range' is shown in Figure 1.2.

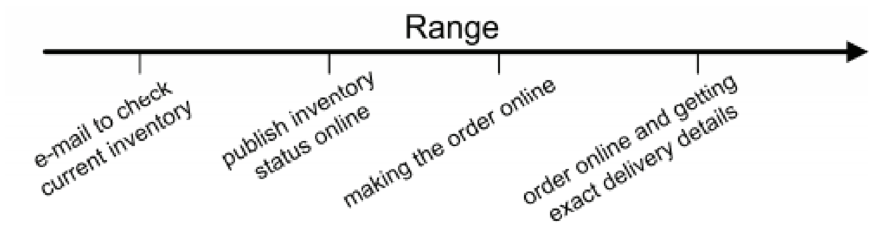
Range
e-mail to check current inventory
publish inventory status online
making the order online
order online and getting exact delivery details

Figure 1.2: Example to show varying Range from [Kee91; WB98]

In the Figure 1.2, the level of granularity increases as the functionality moves from 'accessing e-mail message' to 'publishing status online' and then to 'creating order'. It is due to the change in the amount of data access involved in each kind of functionality. Thus, the 'Range' directly depends upon the range of data access. As the service grows, alongside 'Reach' and 'Range' also peaks up, which means the extent of consumers as well as the kind of functionality increase. This will add complexity in the service. The solution proposed by [Coc01] is to divide the architecture into services. However, only 'Reach' and 'Range' will not be enough to define the service. It will be equally important to determine scope of the individual services. The functionality of the service then should be define in two distinct dimensions 'which kind of functionality' and 'how much functionality'. This leads to another dimension of service as described below. [Kee91; WB98; RS07]

3. Realm: It tries to create a boundary around the scope of the functionality provided by the service and thus clarifies the ambiguity created by 'Range'. If we take the same example as shown by Figure 1.2, the range alone defining the kind of functionaly such as creating online order does not explicitly clarifies about what kind of order is under consideration. The order can be customer order or sales order. The specification of 'Realm' defining what kind of order plays role here. So, we can have two different services each with same 'Reach' and 'Range' however different 'Realm' for customer order and Sales order. [Kee91; WB98; RS07]

The consideration of all aspects of a service including 'Reach', 'Range' and 'Realm' give us a model to define granularity of a service and is called R^3 model. The volume in the R^3 space for a service gives its granularity. A coarse-grained service has higher R^3 volume then fine-grained service. The Figure 1.3 and Figure 1.4 show such volume-granularity analogy given by R^3 model. [Kee91; WB98; RS07]

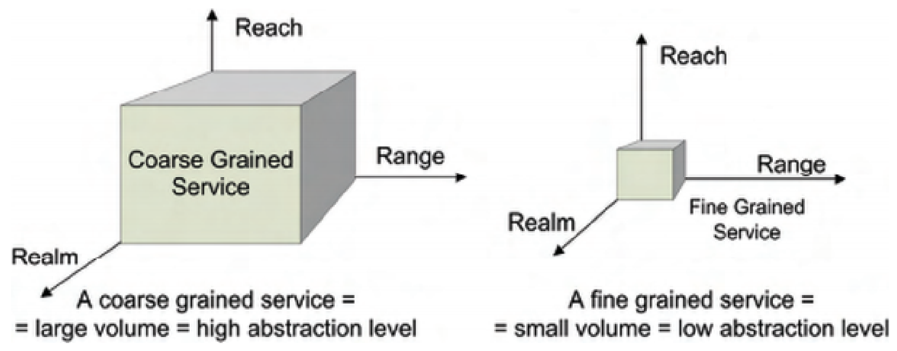
Reach
Range
Realm
Coarse Grained Service
A coarse grained service =
= large volume = high abstraction level
Reach
Range
Realm
Fine Grained Service
A fine grained service =
= small volume = low abstraction level

Figure 1.3: R^3 Volume-Granularity Analogy to show direct dependence of granularity and volume [RS07]

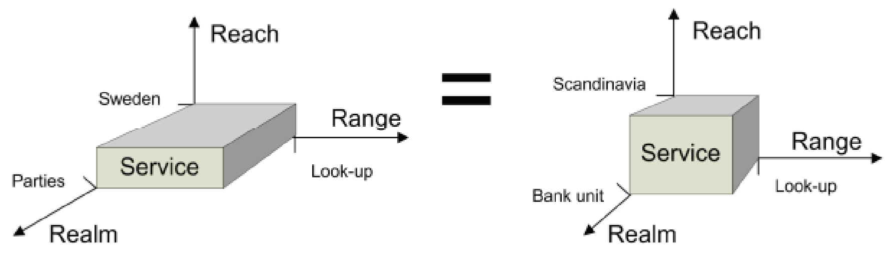

Figure 1.4: R^3 Volume-Granularity Analogy to show same granularity with different dimension along axes [RS07]

1.3.4 Retrospective

The Section 1.3.1 and Section 1.3.2 classified granularity of a service along three different directions: data, functionality and business value. Moreover, the interpretation from the viewpoint of interface by consumer and the viewpoint of producer for realization were also made. Again, the Section 1.3.3 divides the aspect of granularity along Reach, Range and Realm space, the granularity given by the volume in the R^3 space. Despite of good explanation regarding each aspect, the classification along data, functionality and business value do not provide discrete metrics to define granularity in terms of quantitatively. However, the given explanations and criteria are sufficient to compare two different services regarding granularity. These can be effectively used to classify if a service is fine-grained than other service. The R^3 model in Section 1.3.3 additionally

provides various levels of granularity along each axis. This makes it easy to at least measure the granularity and provides flexibility to compare the granularity between various services.
A case study [RS07] using data, functionality and business criteria to evaluate the impact of granularity on the complexity in the architecture is held. The study was performed at KBC Bank Insurance Group of central Europe. Along the study, the impact of each kind of granularity is verified. The important results from the study are listed below.

1. A coarse-grained service in terms of 'Input Data' provides better transactional support, little communication overhead and also helps in scalability. However, there is a chance of data getting out-of-date quickly.

2. A coarse-grained service in terms of 'Output Data' also provides good communication efficiency and supports reusability.

3. A fine-grained service along 'Default Functionality' has high reusability and stability but low reuse efficiency.

4. A coarse-grained service due to more optional functionalities has high reuse efficiency, high reusability and also high stability but the implementation or realization is complicated.

5. A coarse-grained service along 'Business Value' has high consumer satisfaction value and emphasize the architecturally crucial points fulfilling business goals.

Similarly, a study was carried out in Handelsbanken in sweden, which has been working with Service-Oriented architecture. The purpose of the study was to analyze the impact of R^3 model on the architecture. It is observed that there are different categories of functionalities essential for an organization. Some functionality can have high Reach and Range with single functionality and other may have complex functionality with low Reach and Range. The categorization and realization of such functionalities should be accessed individually. It is not necessary to have same level of granularity across all services or there is no one right volume for all services. However, if there are many services with low volume, then we will need many services to accomplish a high level functionality. Additionally, it will increase interdependency among services and network complexity. Finally, the choice of granularity is completely dependent upon the IT-infrastructure of the company. The IT infrastructure decides which level of granularity it can support and how many of them. [RS07]

Principle: IT-infrastructure of an organization affects granularity.
The right value of granularity for an organization is highly influenced by its IT infrastructure. The organization should be capable of handling the complexities such as communication, runtime, infrastructure etc if they choose low granularity. [RS07]

Additionally, it is observed that a single service can be divided into number of granular services by dividing across either Reach, Range or Realm. The basic idea is to make the volume as low as possible as long as it can be supported by IT-infrastructure. Similarly, each dimension is not completely independent from other. For example: If the reach of a service has to be increased from domestic to global in an organization then the realm of the functionality has to be decreased in order to make the volume as low as possible, keep the development and runtime complexities in check. [RS07]

Principle: Keep the volume low if possible.
If supported by IT-infrastructure, it is recommended to keep the volume of service as low as possible, which can be achieved by managing the values of Reach, Range and Realm. It will help to decrease development and maintenance overload. [RS07]

1.4 Problem Statement

This chapter analysed various qualitative aspects related to granularity of microservices. The interesting thing to notice is various dimensions of size across data, functionality and business value. But it is also necessary to look into it in quantitative approach. Additionally, the various principles listed in Section 1.2 points to other qualities of microservices except granularity. The granularity is not only attribute which is important while modeling microservices but there are other quality attributes which affect granularity and influence on the overall quality of microservices. The next task will be to study various quality attributes affecting microservices and derive their quantitative metrics.

2 Quality Attributes of Microservices

This chapter will attempt to find out various factors defining quality of microservices. Also, it will provide quantitative approach to evaluate the overall quality of microservice. At first, the Section 2.2 presents a list of quality attributes compiled from various research papers, which are considered important when defining quality of service. Next, in Section 2.3, various metrices to evaluate these attributes are explained. Furthermore, in Section 2.4, a list of basic metrices being derived from complex metrices introduced in section are created. Again, Section 2.5 lists various principles which shows impact of various attributes on the quality of microservices. Finally, based on the previous result on metrics and principles, relationship amongst quality attributes is tabulated in Section 2.6.

2.1 Introduction

In addition to allign with the business requirements, an important goal of software engineering is to provide high quality. The quality assessment in the context of service oriented product becomes more crucial as the complexity of the system is getting higher by time. [ZL09; GL11; Nem+14] Quality models have been devised over time to evaluate quality of a software. A quality model is defined by quality attributes and quality metrics. Quality attributes are the expected properties of software artifacts defined by the chosen quality model. And, quality metrics give the techniques to measure the quality attributes. [Man+np] The software quality attributes can again be categorized into two types: internal and external attributes. [Man+np; BMB96] The internal quality attributes are the design time attributes which should be fulfilled during the design of the software. Some of the internal quality attributes are loose coupling, cohesion, granularity, autonomy etc.[Ros+11; SSPnp; EM14] On the other hand, external quality attributes are the traits of the software artifacts produced at the end of Software Development Life Cycle Some of them are reusability, maintainability etc. [EM14; Man+np; FL07; Feu13] For that reason, the external quality attributes can only be measured after the end of development. However, it has been evident that internal quality attributes have huge impact upon the value of external quality attributes and thus can be used to predict them. [HS90; Bri+np; AL03; Shi+08]The evaluation of both internal and external quality attributes are valuable in order to produce high quality software.[Man+np; PRF07; Per+07]

2.2 Quality Attributes

As already mentioned in Section 2.1, the internal and external quality attributes determine the overall value of the service getting composed. There are different researches and studies which have been performed to find the factors affecting the service qualities. Based on the published research papers, a comprehensive table 2.1 has been created. The table provides a minimum list of quality attributes which have been considerd in various research papers.

#	Attribute	[SSPnp]	[Xianp]	[AZR11]	[Shi+08]	[Ma+09]	[FL07]
1	Coupling	✓	✓	✓	✓	✓	✓
2	Cohesion	✓	X	✓	✓	✓	✓
3	Autonomy	✓	X	X	X	X	X
4	Granularity	✓	✓	✓	✓	✓	✓
5	Reusability	✓	X	X	✓	X	✓
6	Abstraction	✓	X	X	X	X	X
7	Complexity	X	X	X	✓	X	X

Table 2.1: Quality Attributes

The Table 2.1 gives a picture of the studies performed so far around service quality attributes. It can be deduced that most of the papers focus on coupling and granularity of the service and only few of them focus on other attributes such as complexity and autonomy.

Coupling refers to the dependency and interaction among services. The interaction becomes inevitable when a service requires a functionality provided by another service in order to accomplish its own goals. Similarly, Cohesion of any system is the extent of appropriate arrangement of elements to perform the functionalities. It affects the degree of understandability and sensibility of its interface for its consumers. Whereas, the complexity attribute provides the way to evaluate the difficulty in understanding and reasoning the context of the service or component.[EM14]
The reusability attribute for a service measures the degree to which it can be used by multiple consumer services and can undergo multiple composition to accoplish various high order functionalities. [FL07]
Autonomy is a broad term which refers to the ability of a service for self-containment, self-controlling, self-governance. Autonomy defines the boundary and context of the service. [MLS07] Finally, granularity is described in Chapter 1 in detail. The basic

signifance of granularity is the diversity and size of the functionalities offered by the service. [EM14]

2.3 Quality Metrics

There have been many studies made to define metrics for quality components. A major portion of the research have been made for object oriented and component based development. These metrics are refined to be used in service oriented systems.[Xianp; SSPnp] Firslty, various papers related to quality metrics of service oriented systems are collected. The papers which conducted their findings or proposition based on decent amount of scientific studies and have produced convincing result are only selected.
It is interesting to notice that the evaluation presented in the research papers were only performed using the case studies of their own and not real life scenarios. Also, they have used diversity of equations to define the quality metrics. On the basis of case studies made in the papers only, a fair comparision and choice of only one way to evaluate the quality metrics cannot be made. This is further added by the fact that the few papers cover most of the quality metrics, most of them focus on only few quality metrics such as coupling but not all of them focus on all the quality metrics. Nevertheless, not all of them discuss about the relationship between all quality metrics. [EM14] This creates difficulty to map all the quality metrics into a common calculation family.
Around such situation, this section will focus on facilitating the understanding of the metrics. Despite the case that there is no idea of threshold value of metrics discussed to define the optimal quality level, the discussed method can still be used to evaualte the quality along various metrics and compare the various design artifacts. This will definitely help to choose the optimum design based on the criteria of quality attributes. Also, there is complexity in understanding and confusion to follow a single proposed metrics procudure. But, the existing procedures can be broken down further to the simple understandable terms. By doing so, the basic terms which are the driving factors for the quality attributes and at the same time followed by most of the procedures as defined in the papers, can be identified.
The remaining part of this section will attempt to synthesize the metrics definitions proposed in various papers, analyze them to identify their conceptual base of measurement in simplest form.

2.3.1 Context and Notations

Before looking into the definitions of metrics, it will be helpful to know related terms, assumptions and their respective notations.

- the business domain is realized with various processes defined as $P = \{p_1, p_2 ... p_s\}$
- a set of services realizing the application is defined as $S = \{s_1, s_2 ... s_s\}$; s_s is the total number of services in the application
- for any service $s \epsilon S$, the set of operations provided by s is given by $O(s) = \{o_1, o_2, ... o_o\}$ and $|O(s)|$ = O
- if an operation $o \epsilon O(s)$ has a set of input and output messages given by M(o). Then the set of messages of all operations of the service is given by $M(s) = \cup_{o \epsilon O(s)} M(o)$
- the consumers of the service $s \epsilon S$ is given by $S_{consumer}(s) = \{S_{c1}, S_2, ... Sn_c\}$; n_c gives the number of consumer services
- the producers for the service or the number of services to which the service $s \epsilon S$ is dependent upon is given by $S_{producer}(s) = \{S_{c1}, S_2, ... Sn_p\}$; n_p gives the number of producer services

2.3.2 Coupling Metrics

[SSPnp] defines following metrics to determine coupling

$$SOCI(s) = number\_of\_operations\_invoked$$

$$= |\{o_i \epsilon s_i : \exists_{o \epsilon s} calls(o, o_i) \wedge s \neq s_i\}|$$

$$ISCI(s) = number\_of\_services\_invoked$$

$$= |\{s_i : \exists_{o \epsilon s}, \exists_{o_i \epsilon s_i} calls(o, o_i) \wedge s \neq s_i\}|$$

$$SMCI(s) = size\_of\_message\_required\_from\_other\_services$$

$$= | \cup M(o_i) : (o_i \epsilon s_i) \vee (\exists_{o \epsilon s}, \exists_{o_i \epsilon s_i} calls(o, o_i) \wedge s \neq s_i)|$$

where,

- SOCI is Service Operation Coupling Index
- ISCI is Inter Service Coupling Index
- SMCI is Service Message Coupling Index

- $call(o, o_i)$ represents the call made from service 'o' to service 'o_i'

[Xianp] defines

$$Coupling(S_i) = p \sum_{j=1}^{n_s} (-log(P_L(j))) = \frac{1}{n} \sum_{j=1}^{n_s} (-log(P_L(j)))$$

where,

- n_s is the total number of services connected
- 'n' is the total number of services in the entire application
- $p = \frac{1}{n}$ gives the probability of a service participating in any connection

[Kaz+11] defines

$$Coupling = \frac{dependency\_on\_business\_entities\_of\_other\_services}{number\_of\_operations\_and\_dependencies} = \frac{\sum_{i \epsilon D_s} \sum_{k \epsilon O(s)} CCO(i,k)}{|O(s)|.K}$$

where,

- $D_s = \{D_1, D_2, ...D_k\}$ is set of dependencies the service has on other services
- $K = |D_s|$
- CCO is Conceptual Coupling betweeen service operations and obtained from BE X EBP (CRUD) matrix table constructed with business entities and business operations, where BE is business entity and EBP any logical process defined in an operation.

[Shi+08] defines

$$coupling(s) = \frac{number\_of\_services\_connected}{number\_of\_services} = \frac{n_c + n_p}{n_s}$$

where,

- n_c is number of consumer services
- n_p number of dependent services
- n_s total number of services

[AZR11] defines

$$coupling = \frac{number\_of\_invocation}{number\_of\_services}$$
$$= \frac{\sum_{i=1}^{|O(s)|}(S_{i,sync} + S_{i,async})}{|S|}$$

where,

- $S_{i,sync}$ is the synchronous invocation in the operation O_i
- $S_{i,async}$ is the asynchronous invocation in the operation O_i

The Table 2.2 shows simpler form of metrics proposed for coupling. Although, the table shows different way of evaluating coupling, they somehow agree to the basic metrics to calculate coupling. It can be deduced that the metrics to evaluate coupling uses basic metrics such as number of operations, number of provider services and number of messages.

2.3.3 Cohesion Metrics

[SSPnp] defines following metric for cohesion.

$$SFCI(s) = \frac{number\_of\_operations\_using\_same\_message}{number\_of\_operations}$$
$$= \frac{max(\mu(m))}{|O(s)|}$$

where,

- SFCI is Service Functional Cohesion Index

#	Papers	Metrics Definition
1	[SSPnp]	SOCI : number of operation of other services invoked by the service ISCI : number of services invoked by a service SMCI : total number of messages from information model required by the operations
2	[Xianp]	Coupling is evaluated as information entropy of a complex service component where entropy is calculated using various data such as total number of atomic components connected, total number of links to atomic components and total number of atomic components
3	[Kaz+11]	The coupling is measured by using the dependency of a service operations with operations of other services. Additionally, the dependency is considered to have different weight value for each kind of operation such as Create, Read, Update, Delete (CRUD) and based on the type of business entity involved. The weight is referred from the CRUD matrix constructed in the process.
4	[Shi+08]	evaluated by the average number of directly connected services
5	[AZR11]	defines coupling as the average number of synchronous and asynchronous invocations in all the operations of the service

Table 2.2: Coupling Metrics

- the number of operations using a message 'm' is $\mu(m)$ such that $m \epsilon M(s)$ and |O(s)| > 0
  The service is considered highly cohesive if all the operations use one common message. This implies that a higly cohesive service operates on a small set of key business entities as guided by the messages and are relavant to the service and all the operations operate on the same set of key business entities. [SSPnp]

[Shi+08] defines

$$cohesion = \frac{n_s}{|M(s)|}$$

[PRF07] defines

$$SIDC = \frac{number\_of\_operations\_sharing\_same\_parameter}{total\_number\_of\_parameters\_in\_service}$$

$$= \frac{n_{op}}{n_{pt}}$$

$$SIUC = \frac{sum\_of\_number\_of\_operations\_used\_by\_each\_consumer}{product\_of\_total\_no\_of\_consumers\_and\_operations}$$
$$= \frac{n_{oc}}{n_c * |O(s)|}$$

$$SSUC = \frac{sum\_of\_number\_of\_sequentials\_operations\_accessed\_by\_each\_consumer}{product\_of\_total\_no\_of\_consumers\_and\_operations}$$

$$= \frac{n_{so}}{n_c * |O(s)|}$$

where,

- SIDC is Service Interface Data Cohesion
- SIUC is Service Interface Usage Cohesion
- SSUC is Service Sequential Usage Cohesion
- n_{op} is the number of operations in the service which share the same parameter types
- n_{pt} is the total number of distinct parameter types in the service
- n_{oc} is the total number of operations in the service used by consumers
- n_{so} is the total number of sequentially accessed operations by the clients of the service

[AZR11] defines

$$cohesion = \frac{max(\mu(OFG, ODG))}{|O(s)|}$$

where,

- OFG and ODG are Operation Gunctionality Granularity and Operation Data Granularity respectively as calculated in Section 2.3.4
- $\mu(OFG, ODG)$ gives the number of operations with specific value of OFG and ODG

#	Papers	Metrics Definition
1	[SSPnp]	defines SFCI which measures the fraction of operations using similar messages out of total number of operations in the service operations of the service
2	[PRF07]	SIDC : defines cohesiveness as the fraction of operations based on commonality of the messages they operate on SIUC : defines the degree of consumption pattern of the service operations which is based on the similarity of consumers of the operations SIUC : defines the cohesion based on the sequential consumption behavior of more than one operations of a service by other services
3	[Shi+08]	cohesion is evaluated by the inverse of average number of consumed messages by a service
4	[AZR11]	cohesion is defined as the consensus among the operations of the service regarding the functionality and data granularity which represents the type of parameters and the operations importance as calculated in the Section 2.3.4

Table 2.3: Cohesion Metrics

The Table 2.3 gives simplified view for the cohesion metrics. The papers presented agree on some basic metrics such as similarity of the operation usage behavior, commanility in the messages consumed by operations and similarity in the size of operations.

2.3.4 Granularity Metrics

[SSPnp] defines

$$SCG = number\_of\_operations = |O(s)|$$

$$SDG = size\_of\_messages = |M(s)|$$

where,

- SCG is Service Capability Granularity
- SDG is Service Data Granularity

[Shi+08] defines

$$granularity = \frac{number\_of\_operations}{size\_of\_messages} = \frac{|O(s)|^2}{|M(s)|^2}$$

[AZR11] defines

$$ODG = fraction\_of\_total\_weight\_of\_input\_and\_output\_parameters$$
$$= \left(\frac{\sum_{i=1}^{n_i o} W_{pi}}{\sum_{i=1}^{n_i s} W_{pi}} + \frac{\sum_{j=1}^{n_j o} W_{pj}}{\sum_{j=1}^{n_j s} W_{pj}} \right)$$

$$OFG = complexity\_weightage\_of\_operation = \frac{W_i(o)}{\sum_{i=1}^{|O(S)|} W_i(o)}$$

$$granularity = sum\_of\_product\_of\_data\_and\_functionality\_granularity$$
$$= \sum_{i=1}^{|O(S)|} ODG(i) * OFG(i)$$

where,

- ODG is Operation Data Granularity
- OFG is Operation Functionality Granularity
- W_{pi} is the weight of input parameter
- W_{pj} is the weight of output parameter
- $n_i o$ is the number of input parameters in an operation
- $n_j o$ is the number of output parameters in an operation
- $n_i s$ is the total number of input parameters in the service
- $n_j s$ is the total number of output parameters in the service
- $W_i(o)$ is the weight of an operation of the service
- $|O(S)|$ is the number operations in the service

The Table 2.4 presents various view of granularity metrics based on different research papers. The Chapter 1 discuss in detail regarding the topic. Based on the Table 2.4, the granularity is evaluated using some basic metrics such as the number and type of the parameters, number of operations and number of messages consumed.

#	Papers	Metrics Definition
1	[SSPnp]	the service capability granularity is calculated by the number of operations in a service and the data granularity by the number of messages consumed by the operations of the service
2	[AZR11]	ODG : evaluated in terms of number of input and output parameters and their type such as simple, user-defined and complex OFG : defined as the level of logic provided by the operations of the services SOG : defined by the sum of product of data granularity and functionality granularity for all operations in the service
3	[Shi+08]	evaluated as the ratio of squared number of operations of the service to the squared number of messages consumed by the service

Table 2.4: Granularity Metrics

2.3.5 Complexity Metrics

[ZL09] defines

$$RIS = \frac{IS(s_i)}{|S|}$$

$$RCS = \frac{coupling\_of\_service}{number\_of\_services} = \frac{CS(s_i)}{|S|}$$

where,

- $CS(s_i)$ gives coupling value of a service
- |S| is the number of services to realize the application
- $IS(s_i)$ gives importance weight of a service in an application

The complexity is rather obtained by relative coupling than by coupling on its own. A low value of RCS indicates that the coupling is lower than the count of services where as RCS with value 1 indicates that the coupling is equal to the number of services. This represents high amount of complexity.
Similarly, a high value of RIS indicates that a lot of services are dependent upon the service and the service is of critical value for other services. This increases complexity

as any changes or problem in the service affects a large number of services to a high extent.
[AZR11] defines

$$Complexity(s) = \frac{Service\_Granularity}{number\_of\_services}$$
$$= \frac{\sum_{i=1}^{|O(s)|}(SG(i))^2}{|S|}$$

where,

- |S| is the number of services to realize the application
- SG(i) gives the granularity of ith service
  calculated as described in Section 2.3.4

#	Papers	Metrics Definition
1	[ZL09]	RCS : complexity evaluated by the degree of coupling for a service and evaluated as the fraction of its coupling to the total number of services RIS : measured as the fraction of total dependency weight of consumers upon the service to the total number of services
2	[AZR11]	the complexity is calculated using the granunarity of service operations

Table 2.5: Complexity Metrics

The Table 2.5 provides the way to interprete complexity metrics. The complexity is highly dependent upong coupling and functionality granularity of the service.

2.3.6 Autonomy Metrics

[Ros+11] defines

$$Self-Containment(SLC) = sum\_of\_CRUD\_coefficients\_for\_each\_Business\_entity$$
$$= \frac{1}{h_2 - l_2 + 1}\sum_{i=l_2}^{h_2}\sum_{sr \epsilon SR}(BE_{i,sr}XV_{sr})$$

$$Dependency(DEP) = dependency\_of\_service\_on\_other\_service\_business\_entities$$
$$= \frac{\sum_{j=L1_i}^{h1_i} \sum_{k=1}^{BE} V_{sr_{jk}} - \sum_{j=L1_i}^{h1_i} \sum_{k=L2_i}^{j2_i} V_{sr_{jk}}}{nc}$$

$$\text{autonomy} = \begin{cases} SLC - DEP & \text{if SLC > DEP} \\ 0 & \text{otherwise} \end{cases}$$

where,

- nc is the number of relations with other services
- $BE_{i,sr} = 1$ if the service performs action sr on the ith BE and sr represents any CRUD operation and V_{sr} is the coefficient depending upon the type of action
- $V_{sr_{jk}}$ is the corresponding value of the action in jkth element of CRUD matrix, it gives the weight of corresponding business capability affecting a business entity
- $l1_i, h1_i, l2_i, h2_i$ are bounding indices in CRUD matrix of ith service

#	Papers	Metrics Definition
1	[Ros+11]	SLC : defined as the degree of control of a service upon its operations to act on its Business entities only DEP : given by the degree of coupling of the service with other services autonomy is given by the difference of SLC and DEP if SLC > DEP else it is taken as 0

Table 2.6: Autonomy Metrics

The Table 2.6 shows a way to interprete autonomy. It is calculated by the difference SLC-DEP when SLC is greater than DEP. In other cases it is taken as zero. So, autonomy increases as the operations of the services have full control upon its business entities but decreases if the service is dependent upon other services.

2.3.7 Reusability Metrics

[SSPnp] defines

$$\begin{aligned} Reusability &= number\_of\_existing\_consumers \\ &= |S_{consumers}| \end{aligned}$$

[Shi+08] defines

$$Reusability = \frac{Cohesion - granularity + Consumability - coupling}{2}$$

#	Papers	Metrics Definition
1	[SSPnp]	SRI defines reusability as the number of existing consumers of the service
2	[Shi+08]	evaluated from coupling, cohesion, granularity and consumability of a service where consumability is the chance of the service being discovered and depends upon the fraction of operations in the service

Table 2.7: Reusability Metrics

The table 2.7 shows the reusability metrics evaluation. Reusability depends upon coupling, cohesion and granularity. It decreases as coupling and granularity increases.

2.4 Basic Quality Metrics

The Section 2.3 presents various metrics to evaluate quality attributes based upon different papers. Additionally, the section analyzed the metrics and tried to derive them in simplest form possible. Eventually, the tables demonstrating the simplest definition of the metrics shows that the different quality attributes are measured on the basis of some basic metrics. Based on the Section 2.3 and based on the papers, this section will attempt to derive basic metrics. The Table 2.8 provides a list of basic metrices.

Considering the Table 2.8, it can be deduced that cohesion of a service increases with the number of operations using similar messages, using similar parameters and serving same consumer. The same process can be followed for other quality attributes.

#	Metrics	Coupling	Cohesion	Granularity	Complexity	Autonomy	Reusability
1	number of service operations invoked by the service	+			+	-	-
2	number of operation using similar messages		+				+
3	number of operation used by same consumer		+				+
4	number of operation using similar parameters		+				+
5	number of operation with similar scope or capability		+				+
6	scope of operation			+	+		-
7	number of operations			+	+		-
8	number of parameters in operation			+	+		-
9	type of parameters in operation			+	+		-
10	number and size of messages used by operations	+		+	+	-	-
11	type of messages used by operations		+				+
12	number of consumer services	+			+	-	+
13	number of producer services	+			+	-	-
14	type of operation and business entity invoked by the service	+			+	-	-
15	number of consumers accessing same operation		+				+
16	number of consumer with similar operation usage sequence		+				+
17	dependency degree or imporance of service operation to other service				+		
18	degree of control of operation to its business entities					+	

Table 2.8: Basic Quality Metrics

Additionally, the effect of the basic metrics upon various quality attributes are also evaluated and stated under each column such as 'coupling', 'cohesion' etc. Here, the meaning of the various symbols to demonstrate the affect are as listed below:

- + the basic metric affect the quality attribute proportionlly
- - the basic metric inversely affect the quality attribute
- there is no evidence found regarding the relationship from the papers

2.5 Principles defined by Quality Attributes

The Section 2.1 has already mentioned that there are two distinct kind of quality attributes: external and internal. The external quality attributes cannot be evaluated during design however can be predicted using the internal quality attributes. This kind of relationship can be used to design service with good quality. Moreover, it is important to identify how each internal attributes affect the external attributes. The understanding of such relationship will be helpful to determine the combined effect of the quality attributes and to identify the service with appropriate level of quality based on the requirement. The remaining part of the section lists relationship existing in various quality attributes as well as the desired value of the quality attributes.

1. When a service has large number of operations, it means it has large number of consumers. It will highly affect maintainability because a small change in the operations of the service will be propagated to large number of consumers. But again, if the service is too fine granular, there will be high network coupling. [FL07; Xianp][BKM07]
2. Low coupling improves understandability, reusability and scalability where as high coupling decreases maintainability. [Kaz+11][Erl05][Jos07]
3. A good strategy for grouping operations to realize a service is to combine the ones those are used together. This indicates that most of the operations are used by same client. It highly improves maintainability by limiting the number of affected consumer in the event of change in the service. [Xianp]
4. A high cohesive quality attribute defines a good service. The service is easy to understand, test, change and is more stable. These properties highly support maintainability. enumerate[np01]
5. A service with operations those are cohesive and have low coupling, make the service reusable. [WYF03][FL07][Ma+09]
6. Services must be selected in a way so that they focus on a single business functionality. This highly follows the concept of low coupling. [PRF07][SSPnp]
7. Maintainability is divided into four distinct concepts: analyzability, changeability, stability and testability.[np01] A highly cohesive and low-coupled design should be easy to analyze and test. Moreover, such a system will be stable and easy to change.[PRF07]
8. The complexity of a service is determined by granularity. A coarse-grained service has higher complexity. However, as the size of the service decreases, the complexity of the over system governance also increases. [AZR11]
9. The complexity depends upon coupling. The complexity of a service is defined in terms of the number of dependencies of a service, number of operations as well as number and size of messages used by the service operations.[AZR11][SSPnp][Lee+01]

10. The selection of an appropriate service has to deal with multi-objective optimization problem. The quality attributes are not independent in all aspects. Depending upon goals of the architecture, tradeoffs have to be made for mutually exclusive attributes. For example, when choosing between coarse-grained and fine-grained service, various factors such as governance, high network roundtrip etc. also should be considered. [JSM08]

11. Business entity convergence of a service, which is the degree of control over specific business entities, is an important quality for the selection of service. For example: It is be better to create a single service to create and update an order. In that way change and control over the business entity is localized to a single service.[Ma+09]

12. Increasing the granularity decreases cohesion but also decreases the amount of message flow across the network. This is because, as the boundary of the service increases, more communication is handled within the service boundary. However, this can be true again only if highly related operations are merged to build the service. This suggests for the optimum granularity by handling cohesion and coupling in an appropriate level.[Ma+09][BKM07]

13. As the scope of the functionality provided by the service increases, the reusability decreases. [FL07]

14. If the service has high interface granularity, the operations operates on coarse-grained messages. This can affect the service performance. On the other hand, if the service has too fine-grained interface, then there will be a lot of messages required for the same task, which then can again affect the overall performance. [BKM07]

2.6 Relationship among Quality Attributes

In order to determine the appropriate level of quality for a service, it is also important to know the relationship between the quality attributes. This knowledge will helpfull to decide tradeoffs among them in the situation when it is not possible to achieve best of all. Based on the Section 2.2 and Section 2.5, the identified relationship among the quality attributes are shown in the Table 2.9.

Here, the meaning of the various symbols showing nature of relationship are as listed below:

#	Quality Attributes	Coupling	Cohesion	Granularity
1	Coupling		-	+
2	Cohesion	-		
3	Granularity	+		
4	Complexity	+		+
5	Reusability	-	+	-
6	Autonomy	-		
7	Maintainability	-	+	-

Table 2.9: Relationship among quality attributes

- + the quality attribute on the column affects the quality attribute on the corresponding row positively
- - the quality attribute on the column affects the quality attribute on the corresponding row inversely
- there is no enough evidence found regarding the relationship from the papers or these are same attributes

Based on the Table 2.9, if granularity of a microservice increases then coupling of other microservices on it, will also increase. Similarly, relationship among other quality attributes can also be derived using the table. The idea regarding impact of various quality attributes amongst each other can be helpful to make decision regarding trade offs.

2.7 Conclusion

There is no doubt that granularity is an important aspect of a microservice however there are other different factors which affect granularity as well as the overall quality of the service. Again, knowing the way to evaluate these qualities in terms of a quantitative figure can be helpful for easy decision regarding quality. However, most of the metrices are quite complex so the Section 2.4 compiled these complex mectrics in terms of simple metrices. The factors used to define these basic metrices are the kind which are accessible to normal developers. So, these basic metrices can be an efficient and easy way to determine quality of microservices.
Moreover, the external quality attributes such as reusability, scalability etc can be controlled by fixing internal quality attributes such as coupling, cohesion, autonomy etc. So, finding an easy way to evaluate and fix internal quality attributes will eventually

make it easier to achieve microservices with satisfactory value of external quality attributes.
Nevertheless, the quality attributes are not mutually exclusive but are dependent on each other. The Table 2.9 provides basic relationship among them. This kind of table can be helpful when needed to perform trade-offs among various attributes depending upon goals.

2.8 Problem Statement

Having collected important concepts regarding various quality attributes, which is one of the major drivers for defining architecture as mentioned in Section **??**, the next important concept is to find the process of modeling microservices as stated in the Table **??**. The quality attributes will be a major input for deciding the process of identifying microservices from a problem domain.

List of Figures

List of Tables

Bibliography

[AL03] M. Alshayeb and W. Li. *An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes*. Tech. rep. IEEE Computer Society, 2003.

[AZR11] S. Alahmari, E. Zaluska, and D. C. D. Roure. *A Metrics Framework for Evaluating SOA Service Granularity*. Tech. rep. School of Electronics and Computer Science University Southampton, 2011.

[BKM07] P. Bianco, R. Kotermanski, and P. F. Merson. *Evaluating a Service-Oriented Architecture*. Tech. rep. Carnegie Mellon University, 2007.

[BMB96] L. C. Briand, S. Morasca, and V. R. Basili. *Property-Based Software Engineering Measurement*. Tech. rep. IEEE Computer Society, 1996.

[Bri+np] L. C. Briand, J. Daly, V. Porter, and J. Wüst. *A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems*. Tech. rep. Fraunhofer IESE, np.

[Coc01] A. Cockburn. *WRITING EFFECTIVE USE CASES*. Tech. rep. Addison-Wesley, 2001.

[EM14] A. A. M. Elhag and R. Mohamad. *Metrics for Evaluating the Quality of Service-Oriented Design*. Tech. rep. Universiti Teknologi Malaysia, 2014.

[Erl05] T. Erl. *Service-Oriented Architecture Concepts, Technology, and Design*. Prentice Hall Professional Technical Reference, 2005.

[Feu13] G. Feuerlicht. *Evaluation of Quality of Design for Document-Centric Software Services*. Tech. rep. University of Economics and University of Technology, 2013.

[FL07] G. Feuerlicht and J. Lozina. *Understanding Service Reusability*. Tech. rep. University of Technology, 2007.

[Foo05] D. Foody. *Getting web service granularity right*. 2005.

[GL11] A. Goeb and K. Lochmann. *A software quality model for SOA*. Tech. rep. Technische Universität München and SAP Research, 2011.

[Hae+np] R. Haesen, M. Snoeck, W. Lemahieu, and S. Poelmans. *On the Definition of Service Granularity and Its Architectural Impact*. Tech. rep. Katholieke Universiteit Leuven, np.

[HS00] P. Herzum and O. Sims. *Business Components Factory: A Comprehensive Overview of Component-Based Development for the Enterprise*. John Wiley Sons, 2000.

[HS90] S. Henry and C. Selig. *Predicting Source=Code Complexity at the Design Stage*. Tech. rep. Virginia Polytechnic lnstirure, 1990.

[Jos07] N. M. Josuttis. *SOA in Practice The Art of Distributed System Design*. O'Reilly Media, 2007.

[JSM08] P. Jamshidi, M. Sharifi, and S. Mansour. *To Establish Enterprise Service Model from Enterprise Business Model*. Tech. rep. Amirkabir University of Technology, Iran University of Science, and Technology, 2008.

[Kaz+11] A. Kazemi, A. N. Azizkandi, A. Rostampour, H. Haghighi, P. Jamshidi, and F. Shams. *Measuring the Conceptual Coupling of Services Using Latent Semantic Indexing*. Tech. rep. Automated Software Engineering Research Group, 2011.

[Kee91] P. G. Keen. *Shaping The Future of Business Design Through Information Technology*. Harvard Business School Press, 1991.

[Lee+01] J. K. Lee, S. J. Jung, S. D. Kim, W. H. Jang, and D. H. Ham. *Component Identification Method with Coupling and Cohesion*. Tech. rep. Soongsil University and Software Quality Evaluation Center, 2001.

[Linnp] D. Linthicum. *Service Oriented Architecture (SOA)*. np.

[Ma+09] Q. Ma, N. Zhou, Y. Zhu, and H. Wang1. *Evaluating Service Identification with Design Metrics on Business Process Decomposition*. Tech. rep. IBM China Research Laboratory and IBM T.J. Watson Research Center, 2009.

[Man+np] M. Mancioppi, M. Perepletchikov, C. Ryan, W.-J. van den Heuvel, and M. P. Papazoglou. *Towards a Quality Model for Choreography*. Tech. rep. European Research Institute in Services Science, Tilburg University, np.

[MLS07] Y.-F. Ma, H. X. Li, and P. Sun. *A Lightweight Agent Fabric for Service Autonomy*. Tech. rep. IBM China Research Lab and Bei Hang University, 2007.

[Nem+14] H. Nematzadeh, H. Motameni, R. Mohamad, and Z. Nematzadeh. *QoS Measurement of Workflow-Based Web Service Compositions Using Colored Petri Net*. Tech. rep. Islamic Azad University Sari Branch and Universiti Teknologi Malaysia, 2014.

[np01] np. *ISO/IEC 9126-1 Software Engineering Product Quality – Quality Model*. Tech. rep. International Standards Organization, 2001.

[Per+07] M. Perepletchikov, C. Ryan, K. Frampton, and Z. Tari. *Coupling Metrics for Predicting Maintainability in Service-Oriented Designs*. Tech. rep. RMIT University, 2007.

[PRF07] M. Perepletchikov, C. Ryan, and K. Frampton. *Cohesion Metrics for Predicting Maintainability of Service-Oriented Software*. Tech. rep. RMIT University, 2007.

[RKKnp] C. Rolland, R. S. Kaabi, and N. Kraiem. *On ISOA: Intentional Services Oriented Architecture*. Tech. rep. Université Paris, np.

[Ros+11] A. Rostampour, A. Kazemi, F. Shams, P. Jamshidi, and A. Azizkandi. *Measures of Structural Complexity and Service Autonomy*. Tech. rep. Shahid Beheshti University GC, 2011.

[RS07] P. Reldin and P. Sundling. *Explaining SOA Service Granularity– How IT-strategy shapes services*. Tech. rep. Linköping University, 2007.

[Shi+08] B. Shim, S. Choue, S. Kim, and S. Park. *A Design Quality Model for Service-Oriented Architecture*. Tech. rep. Sogang University, 2008.

[SSPnp] R. Sindhgatta, B. Sengupta, and K. Ponnalagu. *Measuring the Quality of Service Oriented Design*. Tech. rep. IBM India Research Laboratory, np.

[WB98] P. Weill and M. Broadbent. *Leveraging The New Infrastructure: How Market Leaders Capitalize on Information Technology*. Harvard Business School Press, 1998.

[WV04] L. Wilkes and R. Veryard. "Service-Oriented Architecture: Considerations for Agile Systems." In: *np* (2004).

[WYF03] H. Washizakia, H. Yamamoto, and Y. Fukazawa. *A metrics suite for measuring reusability of software components*. Tech. rep. Waseda University, 2003.

[Xianp] W. Xiao-jun. *Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture*. Tech. rep. Nanjing University of Posts and Telecommunications, np.

[ZL09] Q. Zhang and X. Li. *Complexity Metrics for Service-Oriented Systems*. Tech. rep. Hefei University of Technology, 2009.