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:	

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, Rajendra Kharbuja

Acknowledgments

Abstract

The microservices architecture provides various advantages such as agility, independent scalability etc., as compared to the monolithic 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 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 1.2 presents a list of quality attributes compiled from various research papers, which are considered important when defining quality of service. Next, in Section 1.3, various metrices to evaluate these attributes are explained. Furthermore, in Section 1.4, a list of basic metrices being derived from complex metrices introduced in section are created. Again, Section 1.5 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 1.6.

1.1 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. [**Zhang:2009aa**; **Goeb:2011aa**; **Nematzadeh:2014aa**] 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. [**Mancioppi:2015aa**] The software quality attributes can again be categorized into two types: internal and external attributes. [**Mancioppi:2015aa**; **Briand:1996aa**] 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.[**Rostampour:2011aa**; **Sindhgatta:2015aa**; **Elhag:2014aa**] 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. [**Elhag:2014aa**; **Mancioppi:2015aa**; **Feuerlicht:2007aa**; **Feuerlicht:2013aa**] 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. [**Henry:1990aa**;

Briand:2015aa; **Alshayeb:2003aa**; **Bingu-Shim:2008aa**]The evaluation of both internal and external quality attributes are valuable in order to produce high quality software.[**Mancioppi:2015aa**; **Perepletchikov:2007aa**; **Mikhail-Perepletchikov:2015aa**]

1.2 Quality Attributes

As already mentioned in Section 1.1, 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 1.1 has been created. The table provides a minimum list of quality attributes which have been considerd in various research papers.

#	Attribute	[Sindhgatta:2015aa]	[Xiao-jun:2015aa]	[Saad-Alahmari:2011aa]	[Bingu-Shim:2008aa]	[Ma:2009aa]	[Feuerlicht:2007aa]
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 1.1: Quality Attributes

The Table 1.1 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.[**Elhag:2014aa**]
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. [**Feuerlicht:2007aa**]
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. [**Ma:2007aa**] Finally, granularity is described in Chapter **??** in detail. The basic signifance of granularity is the diversity and size of the functionalities offered by the service. [**Elhag:2014aa**]

1.3 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.[**Xiao-jun:2015aa**; **Sindhgatta:2015aa**] 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. [**Elhag:2014aa**] 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.

1.3.1 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 $S_{consumer}(s) = \{S_{c1}, S_2, ... S n_c\}$; 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 $S_{producer}(s) = \{S_{c1}, S_2, ... S n_p\}$; n_p gives the number of producer services

1.3.2 Coupling Metrics

[**Sindhgatta:2015aa**] 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'

[**Xiao-jun:2015aa**] 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

[**Kazemi:2011aa**] 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.

[**Bingu-Shim:2008aa**] 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

[**Saad-Alahmari:2011aa**] 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 1.2 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.

1.3.3 Cohesion Metrics

[**Sindhgatta:2015aa**] 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	[**Sindhgatta:2015aa**]	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	[**Xiao-jun:2015aa**]	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	[**Kazemi:2011aa**]	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	[**Bingu-Shim:2008aa**]	evaluated by the average number of directly connected services
5	[**Saad-Alahmari:2011aa**]	defines coupling as the average number of synchronous and asynchronous invocations in all the operations of the service

Table 1.2: 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. **[Sindhgatta:2015aa]**

[Bingu-Shim:2008aa] defines

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

[Perepletchikov:2007aa] 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

[Saad-Alahmari:2011aa] 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 1.3.4
- $\mu(OFG, ODG)$ gives the number of operations with specific value of OFG and ODG

#	Papers	Metrics Definition
1	**[Sindhgatta:2015aa]**	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	**[Perepletchikov:2007aa]**	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	**[Bingu-Shim:2008aa]**	cohesion is evaluated by the inverse of average number of consumed messages by a service
4	**[Saad-Alahmari:2011aa]**	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 1.3.4

Table 1.3: Cohesion Metrics

The Table 1.3 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.

1.3.4 Granularity Metrics

[Sindhgatta:2015aa] defines

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

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

where,

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

[Bingu-Shim:2008aa] defines

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

[Saad-Alahmari:2011aa] 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 1.4 presents various view of granularity metrics based on different research papers. The Chapter **??** discuss in detail regarding the topic. Based on the Table 1.4, 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	**[Sindhgatta:2015aa]**	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	**[Saad-Alahmari:2011aa]**	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	**[Bingu-Shim:2008aa]**	evaluated as the ratio of squared number of operations of the service to the squared number of messages consumed by the service

Table 1.4: Granularity Metrics

1.3.5 Complexity Metrics

[Zhang:2009aa] 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.
[Saad-Alahmari:2011aa] 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 1.3.4

#	Papers	Metrics Definition
1	**[Zhang:2009aa]**	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	**[Saad-Alahmari:2011aa]**	the complexity is calculated using the granunarity of service operations

Table 1.5: Complexity Metrics

The Table 1.5 provides the way to interprete complexity metrics. The complexity is highly dependent upong coupling and functionality granularity of the service.

1.3.6 Autonomy Metrics

[Rostampour:2011aa] 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	[**Rostampour:2011aa**]	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 1.6: Autonomy Metrics

The Table 1.6 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.

1.3.7 Reusability Metrics

[**Sindhgatta:2015aa**] defines

$$Reusability = number\_of\_existing\_consumers$$
$$= |S_{consumers}|$$

[**Bingu-Shim:2008aa**] defines

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

#	Papers	Metrics Definition
1	[**Sindhgatta:2015aa**]	SRI defines reusability as the number of existing consumers of the service
2	[**Bingu-Shim:2008aa**]	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 1.7: Reusability Metrics

The table 1.7 shows the reusability metrics evaluation. Reusability depends upon coupling, cohesion and granularity. It decreases as coupling and granularity increases.

1.4 Basic Quality Metrics

The Section 1.3 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 1.3 and based on the papers, this section will attempt to derive basic metrics. The Table 1.8 provides a list of basic metrices.

Considering the Table 1.8, 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 1.8: 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

1.5 Principles defined by Quality Attributes

The Section 1.1 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. [**Feuerlicht:2007aa**; **Xiao-jun:2015aa**][**Bianco:2007aa**]

2. Low coupling improves understandability, reusability and scalability where as high coupling decreases maintainability. [**Kazemi:2011aa**][**Erl:2005aa**][**Josuttis:2007aa**]

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. [**Xiao-jun:2015aa**]

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[**np:2001aa**]

5. A service with operations those are cohesive and have low coupling, make the service reusable. [**Washizakia:2003aa**][**Feuerlicht:2007aa**][**Ma:2009aa**]

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. [**Perepletchikov:2007aa**][**Sindhgatta:2015aa**]

7. Maintainability is divided into four distinct concepts: analyzability, changeability, stability and testability.[**np:2001aa**] 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.[**Perepletchikov:2007aa**]

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. [**Saad-Alahmari:2011aa**]

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.[**Saad-Alahmari:2011aa**][**Sindhgatta:201**

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. [**Jamshidi:2008aa**]

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:2009aa**]

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:2009aa**][**Bianco:2007aa**]

13. As the scope of the functionality provided by the service increases, the reusability decreases. [**Feuerlicht:2007aa**]

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. [**Bianco:2007aa**]

1.6 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 1.2 and Section 1.5, the identified relationship among the quality attributes are shown in the Table 1.9.

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

- + the quality attribute on the column affects the quality attribute on the corresponding row positively

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

Table 1.9: Relationship among quality attributes

- - 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 1.9, 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.

1.7 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 1.4 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 1.9 provides basic relationship among them. This kind of table can be helpful when needed to perform trade-offs among various attributes depending upon goals.

1.8 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

[Abr14] S. Abram. *Microservices*. Oct. 2014. URL: http://www.javacodegeeks.com/2014/10/microservices.html.

[Ann14] R. Annett. *What is a Monolith?* Nov. 2014.

[Bro15] S. Brown. "Software Architecture for Developers." In: (Dec. 2015).

[Coc15] A. Cockcroft. *State of the Art in Mircroservices*. Feb. 2015. URL: http://www.slideshare.net/adriancockcroft/microxchg-microservices.

[DMT09] E. M. Dashofy, N. Medvidovic, and R. N. Taylor. *Software Architecture: Foundations, Theory, and Practice*. John Wiley Sons, Jan. 2009.

[FA15] M. T. Fisher and M. L. Abbott. *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise, Second Edition*. Addison-Wesley Professional, 2015.

[FL14] M. Fowler and J. Lewis. *Microservices*. Mar. 2014. URL: http://martinfowler.com/articles/microservices.html.

[Gup15] A. Gupta. *Microservices, Monoliths, and NoOps*. Mar. 2015.

[Mac14] L. MacVittie. *The Art of Scale: Microservices, The Scale Cube and Load Balancing*. Nov. 2014.

[New15] S. Newman. *Building Microservices*. O'Reilly Media, 2015.

[np07] np. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Tech. rep. Keele University, 2007.

[NS14] D. Namiot and M. Sneps-Sneppe. *On Micro-services Architecture*. Tech. rep. Open Information Technologies Lab, Lomonosov Moscow State University, 2014.

[Ric14a] C. Richardson. *Microservices: Decomposing Applications for Deployability and Scalability*. May 2014.

[Ric14b] C. Richardson. *Pattern: Microservices Architecture*. 2014.

[Ric14c] C. Richardson. *Pattern: Monolithic Architecture*. 2014. URL: http://microservices.io/patterns/monolithic.html.

[RTS15] G. Radchenko, O. Taipale, and D. Savchenko. *Microservices validation: Mjolnirr platformcase study*. Tech. rep. Lappeenranta University of Technology, 2015.

[Woo14] B. Wootton. *Microservices - Not A Free Lunch!* Apr. 2014. URL: `http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html`.