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

Contents

1 Granularity

1.1 Introduction

The granularity of a service is often ambiguous and has different interpretation. In simple term, 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 vague and 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 Related Work

A number of researches have been done in the area of service and component granularity. The focus of the researches includes various areas such as definition, effect, its dependent criteria and various measurement metrics.
According to [Pad04], the granularity of a component is inversely related to various non functional qualities such as customization and maintainaibility.
According to Hazen and Sims [HS00], component granularity is rather recursive as a component can be composed of various fine-grained components. They further classify recursion as of discrete or continuous nature but then prefer on discrete recursion. Thus, component granularity can be divide into three discrete layers: system, business and distributed where the composition happens from right to left order and granularity decreases in the opposite order. It is also useful to relate granularity with reusability. Coarse-grained components have high reuse efficiency because they are well focused to a specific business functionality. Whereas fine-grained components have high reusability since they focus on small functionality and thus can be used by other coarse-grained components to accomplish higher level business capabilities. [Mil+01], [WXZ05]
Additionally, Sims [Sim05] gave some insights to measure granularity. The granularity can be measured either a) by the number of components called from an operation on a service interface, b) by the number of components calling the service interface or c) by the number of database tables updated.
Service Oriented Architecture Framework (SOAF [EAK06] also gives some way to measure the granularity in terms of quantitative value. The value can be derived from a) the number of components that are invoked by the service interface and b) the number of changes of states in resources like the database tables updates influenced by the service. Again, the granularity can affect reusability and the network communication to accomplish any task. Coarse grained components have more volume of data exchanged where as fine grained components have high frequency of round trips to complete the full business functionality. It is recommended to balance them which can depend upon various factors such as abstraction level, change expectation rate, service complexity, desired coupling and cohesion. Furthermore, the granularity can be at different levels within an application. High-level business process can be realized as a coarse-grained services and each coarse-grained component can be composed of various fine-grained services.
The idea of multi-layered granularity is further supported by Security Trading case study [EAK06] and International Financial and Brokerage Services IFBS case study [Nav08]. The design of pilot application for Security Trading consists of various services across various levels such as process, application, shared data and infrastructures and containing services with granularity decreasing from left to the right in the order they

are listed. The application for IFBS is refactored from large number of fine-grained service to multi-layered granular services such as process services, business services, composite services or orchestration services, utility services etc.
A research performed by Steghuis [Ste06] talks about various aspects of granularity such as functionality, flexibility, reusability, complexity, context-independence, Performance, Genericity and sourcing. Some differenciating aspects in comparision to previous researches are context-independence, genericity and sourcing. Context-independence means independence and self sufficiency in terms of data as well as logic. It promotes loose coupling and insists lack of knowledge of state of surrounding and inessential maintainance of own state. Genericity is the quality of making generic services which can be used in different situations such as messaging service. Finally, sourcing is the process of identifying cohesive group of tasks which can be outsourced by considering the coupling associated.

1.3 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 principls 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 vertical 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 upon 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. [Hae+np], [Foo05]

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 level of real world business activities. [RS07]

8. 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]

1.4 Dimensions of Granularity

As already mentioned in 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.

1.4.1 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 achiee 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]

1.4.2 Dimension by Interface Realization

The section 1.4.1 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 1.4.1, 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 done 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]

1.4.3 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

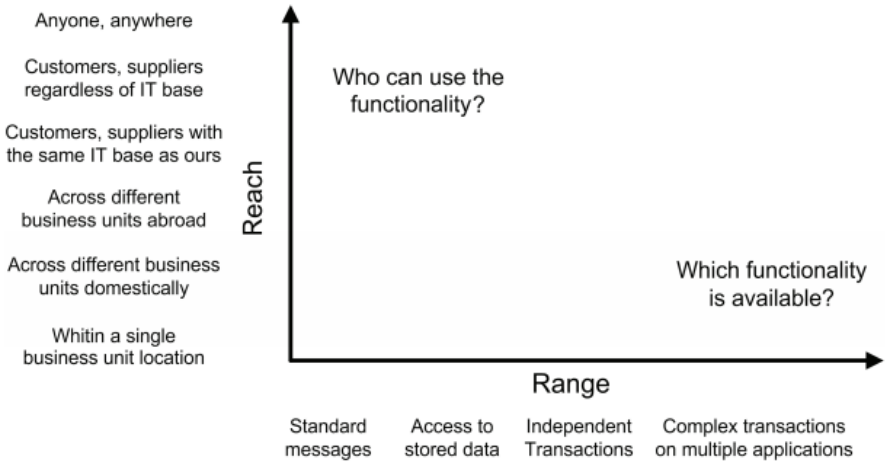

Figure 1.1: Reach and Range model from [Kee91; WB98]

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.
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.

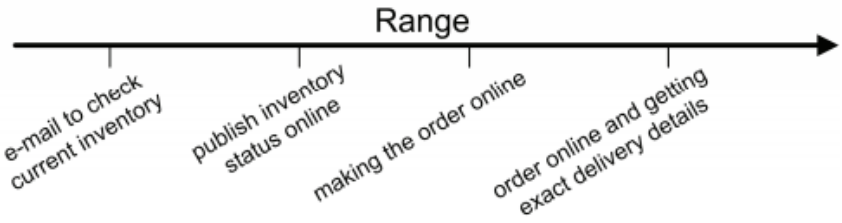

Figure 1.2: Example to show varying Range from [Kee91; WB98]

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]
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 given by Figure **??**, 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 figures **??** and 1.4 show such volume-granularity analogy given by R^3 model. [Kee91; WB98; RS07]

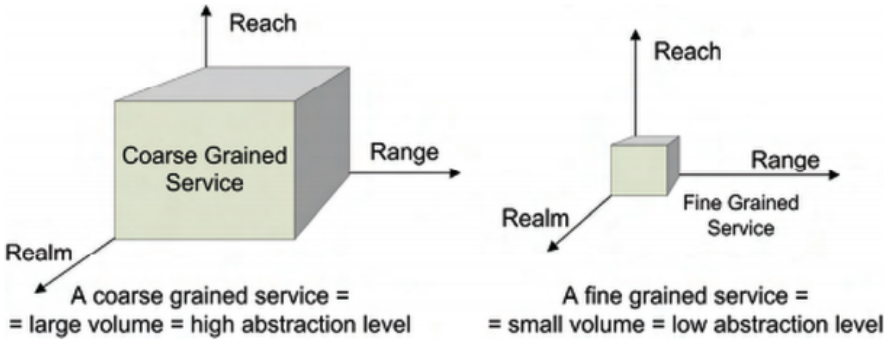

Figure 1.3: R^3 Volume-Granularity Analogy to show direct dependence of granularity and volume [RS07]

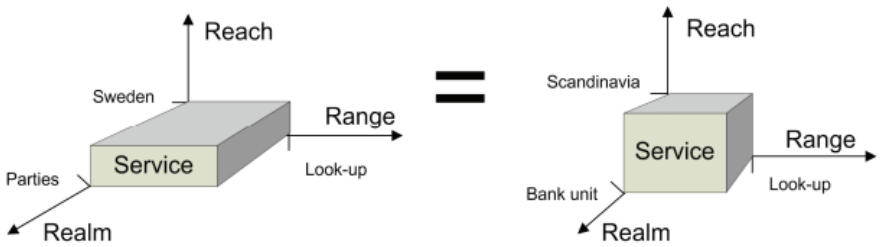

Figure 1.4: R^3 Volume-Granularity Analogy to show same granularity with different dimension along axes [RS07]

1.4.4 Retrospective

The sections 1.4.1 and 1.4.2 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 1.4.3 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 the aspects, 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 well rounded 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 1.4.3 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 done at KBC Bank Insurance Group of central Europe. Along the study, the impact of each kind of granularity is verified. The important results fromt 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]

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 by managing the values of Reach, Range and Realm. It will help to development and maintenance overload. [RS07]

Acronyms

CRUD create, read, update, delete.

IFBS International Financial and Brokerage Services.

SOAF Service Oriented Architecture Framework.

SWIFT Society for Worldwide Interbank Financial Telecommunication.

List of Figures

Bibliography

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

[EAK06] A. Erradi, S. Anand, and N. Kulkarni. *SOAF: An Architectural Framework for Service Definition and Realization*. Tech. rep. University of New South Wales, Infosys Technologies Ltd, 2006.

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

[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.

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

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

[Mil+01] H. Mili, A. Mili, S. Yacoub, and E. Addy. *Reuse-based software engineering: techniques, organization, and controls*. Wiley-Interscience New York, 2001.

[Nav08] V. D. Naveen Kulkarni. *The Role of Service Granularity in A Successful SOA Realization – A Case Study*. Tech. rep. SETLabs, Infosys Technologies Ltd, 2008.

[Pad04] F. Z. Padmal Vitharana Hemant Jain. *Strategy-Based Design of Reusable Business Components*. Tech. rep. IEEE, 2004.

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

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

[Sim05] O. Sims. *Developing the architectural framework for SOA - part 2-service granularity and dependency management. CBDI Forum Journal*. Tech. rep. CBDI, 2005.

[Ste06] C. Steghuis. *Service Granularity in SOA Projects: A Trade-off Analysis*. Tech. rep. University of Twente, 2006.

[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).

[WXZ05] Z. Wang, X. Xu, and D. Zhan. *A Survey of Business Component Identification Methods and Related Techniques*. Tech. rep. International Journal of Information Technology, 2005.