Explaining SOA Service Granularity

– How IT-strategy shapes services

Pierre Reldin and Peter Sundling

Linköping University
INSTITUTE OF TECHNOLOGY

Master Thesis LIU-IEI-TEK-A--07/0090--SE
Department of Management and Engineering
Industrial Economics
Explaining SOA Service Granularity

– How IT-strategy shapes services

Pierre Reldin and Peter Sundling

Supervised by:
Dr. Daniel Kindström – Linköping University
Gunnar Johansson-Hård – Handelsbanken

Linköping University
INSTITUTE OF TECHNOLOGY

Master Thesis LIU-IEI-TEK-A--07/0090--SE
Department of Management and Engineering
Industrial Economics
Today’s competitive business environment forces companies to introduce new product and process innovations at an increasing pace. Almost every aspect of the modern business is supported by information technology systems which, consequently, must evolve at the same pace as the business. A company’s strategic view on IT reflects the strategic importance of IT in the organization, both in terms of the opportunities IT is expected to create and the commitment to IT the business organization is willing to make.

SOA is an emerging concept which aims to structure IT in a more flexible manner. The basic idea is to encapsulate distinct units of business logic in reusable services, which can be combined to support business processes. The term service granularity refers to the amount of logic contained in a service. Even though there is immense hype around SOA today, the concept of service granularity is still relatively unexplored. The service should be coarse grained enough to be reusable, but at the same time specific enough to fit the process. Most SOA literature avoids the subject as being too implementation specific and seldom makes any attempt to concretize the rather abstract term.

The research was conducted at Handelsbanken, which for years has worked with service-oriented principles. The researchers have been given the opportunity to closely analyze the bank’s service initiative. In order to gain an understanding beyond merely technical aspects a rich case study was built, based on interviews with professionals at all levels of the organization.

The research objective was divided in three parts. The first part was to factorize the notion of service granularity, or in other words to find a number of factors which together precisely describe the granularity of a service. The second part was to explicate how the factors are interrelated, i.e. how changing one factor will affect the others. The final part of the objective was to explain how an organization’s strategic view on IT affects the optimal service granularity.

It was found that an organization’s strategic view on IT affects the amount of complexity the organization is able to handle, limiting the optimal SOA granularity, which can be precisely described using three factors: reach, range and realm. Reach defines the locations and people the service is capable of connecting, range defines how much functionality the service offers, and realm defines what kind of functionality the service offers.

Nyckelord/Keyword
service, oriented, architecture, SOA, granularity, IT, strategy, complexity
Abstract

Today’s competitive business environment forces companies to introduce new product and process innovations at an increasing pace. Almost every aspect of the modern business is supported by information technology systems which, consequently, must evolve at the same pace as the business. A company’s strategic view on IT reflects the strategic importance of IT in the organization, both in terms of the opportunities IT is expected to create and the commitment to IT the business organization is willing to make.

SOA is an emerging concept which aims to structure IT in a more flexible manner. The basic idea is to encapsulate distinct units of business logic in reusable services, which can be combined to support business processes. The term service granularity refers to the amount of logic contained in a service. Even though there is immense hype around SOA today, the concept of service granularity is still relatively unexplored. The service should be coarse-grained enough to be reusable, but at the same time specific enough to fit the process. Most SOA literature avoids the subject as being too implementation specific and seldom makes any attempt to concretize the rather abstract term.

The research was conducted at Handelsbanken, which for years has worked with service-oriented principles. The researchers have been given the opportunity to closely analyze the bank’s service initiative. In order to gain an understanding beyond merely technical aspects a rich case study was built, based on interviews with professionals at all levels of the organization.

The research objective was divided in three parts. The first part was to factorize the notion of service granularity, or in other words to find a number of factors which together precisely describe the granularity of a service. The second part was to explicate how the factors are interrelated, i.e. how changing one factor will affect the others. The final part of the objective was to explain how an organization’s strategic view on IT affects the optimal service granularity.

It was found that an organization’s strategic view on IT affects the amount of complexity the organization is able to handle, limiting the optimal SOA granularity, which can be precisely described using three factors: reach, range and realm. Reach defines the locations and people the service is capable of connecting, range defines how much functionality the service offers, and realm defines what kind of functionality the service offers.
Preface

We would like to start this report by expressing our gratitude to Handelsbanken for giving us the opportunity to conduct this research. It has been truly interesting to investigate such a hyped topic as SOA, and elucidate the problem of service granularity. We appreciate the interests all the professionals at Handelsbanken have shown in our work, especially all the architects at the division CDXX. The insights and expertise you have shared with us have challenged us to think in new and constructive ways.

We would in particular like to articulate our gratefulness to our supervisors at the bank Mr. Gunnar Johansson-Hård and Mr. Håkan Sandström. We really appreciate the commitment you have given this project and it has been a true pleasure working with you. Thanks also to all the interviewees who have sacrificed their time, with short notice, in order to share their knowledge with us. The discussions were delightful and necessary for this research. A special thanks goes to Mr. A. H. Johansson, vice president of Handelsbanken, for your time and thoughts.

Particularly valuable input in shaping this report has been contributed by our tutor at Linköping University, Dr. Daniel Kindström, and our opponent Mr. Carl Enoksson. A large thank you goes also to Ms. Georgia Lin and Ms. Margaret Griset for the cumbersome task of reviewing the report.

Finally we would like to express our deepest appreciation for all the love and support from friends and family, especially to Ms. Maria Josefsson and Ms. Georgia Lin. Pierre would furthermore like to dedicate the report in memory of Mr. Karl-Fredrik Hallberg. The nights are no longer so dark, in contrast to a bright future.

Stockholm, March 2007
# Table of Contents

INTRODUCTION ........................................................................................................................................ 1

1  A VISION OF IT MEETING THE NEEDS OF AN EVOLVING BUSINESS .............................................. 1
  1.1  Business strategy and IT support ................................................................................................. 1
  1.2  Challenges related to system integration ..................................................................................... 2
  1.3  Service-Oriented Architecture ..................................................................................................... 2
  1.4  Handelsbanken’s service venture .................................................................................................. 4

2  RESEARCH OBJECTIVES ..................................................................................................................... 4
  2.1  Aim of the investigation .............................................................................................................. 5
  2.2  Scope limitations .......................................................................................................................... 5
  2.2.1  Focus on IT .............................................................................................................................. 5
  2.2.2  Focus on architecture .............................................................................................................. 5
  2.2.3  Focus on service granularity .................................................................................................... 5
  2.2.4  Focus on a single case ............................................................................................................. 5

3  BACKGROUND .................................................................................................................................... 6
  3.1  Introduction to Handelsbanken ..................................................................................................... 6
  3.2  IT at Handelsbanken .................................................................................................................... 6
  3.3  Introduction to Service-Oriented Architecture ............................................................................. 7
  3.4  Handelsbanken’s system for managing Involved Parties ................................................................ 10

4  READING GUIDELINES .......................................................................................................................... 10
  4.1  Structure overview ....................................................................................................................... 11
  4.2  What to read and what not to read ............................................................................................... 12

METHODOLOGY ....................................................................................................................................... 13

5  A SYSTEMS APPROACH WILL BE USED IN POSITIVISTIC MANNER .............................................. 13
  5.1  The main aim of the paper is explanatory .................................................................................. 13
  5.2  The thesis has a positivistic scientific point of departure ........................................................... 13
  5.3  A system view on method will be employed .............................................................................. 14
  5.4  Induction and deduction as different steps of the analytical approach ...................................... 15
  5.5  Systems approach in conjunction with Action Science .............................................................. 16
  5.6  Credibility and objectivity .......................................................................................................... 17

6  INQUIRY APPROACH .......................................................................................................................... 17
  6.1  Qualitative data from case study interviews ............................................................................... 18
  6.2  Both primary and secondary data was used ............................................................................... 18

7  THE WORKING PROCESS ..................................................................................................................... 18
  7.1  Formulating the problem ............................................................................................................ 21
  7.2  Data gathering .............................................................................................................................. 22
  7.3  Analysis and Conclusions ........................................................................................................... 22

LITERATURE REVIEW ............................................................................................................................ 25

8  THREE LEVELS OF OBJECTIVES ..................................................................................................... 25
  8.1  Business perspective on strategic level ...................................................................................... 25
     8.1.1  Business strategy has become concerned with processes .................................................. 25
     8.1.2  Competing on differentiated business processes ............................................................... 27
     8.1.3  Competing on standard processes ....................................................................................... 30
     8.1.4  Core process should be unique, context processes should be standardized .................. 31
  8.2  IT perspective on strategic level ................................................................................................ 31

9  HANDELS BANKEN’S SERVICE VENTURE ............................................................................................ 31
  9.1  Handelsbanken’s system for managing Involved Parties ............................................................ 31
Table of figures

FIGURE 1 EXAMPLE OF FOOD DELIVERY PROCESS ................................................................. 8
FIGURE 2 EXAMPLE OF REUSE IN DRINK DELIVERY PROCESS ........................................... 8
FIGURE 3 SOA MESSAGE EXCHANGE .................................................................................... 9
FIGURE 4 EXAMPLE OF LAYERED ARCHITECTURE (ADAPTED FROM EARL, 2005) ............... 10
FIGURE 5 RESEARCH APPROACHES (ADAPTED FROM ARBNOR & BJERKE, 1997, P. 44-46) .... 14
FIGURE 6 THE ANALYTICAL APPROACH (ADAPTED FROM ARBNOR & BJERKE, 1997, P. 92) ...... 16
FIGURE 7 RESEARCH FRAMEWORK (ADAPTED FROM LEKVALL & WAHLBIN, 2001, P. 183) ........ 19
FIGURE 8 ANALYTICAL APPROACH (ADAPTED FROM ARBNOR & BJERKE, 1997, P. 292) .......... 19
FIGURE 9 REVISED RESEARCH FRAMEWORK ...................................................................... 20
FIGURE 10 SYSTEM ANALYSIS (ADAPTED FROM ARBNOR & BJERKE, 1997, P. 302) ............... 21
FIGURE 11 PYRAMID OF BUSINESS OBJECTIVES ................................................................... 25
FIGURE 12 IT-INFRASTRUCTURE (ADAPTED FROM WEILL AND BROADBENT, 1998, P. 86) ....... 33
FIGURE 13 APPROACHES TO IT-ENABLED BUSINESS TRANSFORMATION (ADAPTED FROM VENKATRAMAN, 1994) ................................................................. 36
FIGURE 14 EXAMPLE OF CHECK BALANCE PROCESS ............................................................ 47
FIGURE 15 REUSED SERVICE IN ANOTHER PROCESS ............................................................ 47
FIGURE 16 EXAMPLE OF SHIPPING PROCESS .......................................................................... 48
FIGURE 17 REACH AND RANGE (ADAPTED FROM KEEN 1991 AND WEILL & BROADBENT,1998) 56
FIGURE 18 EXAMPLE OF INCREASING RANGE ...................................................................... 57
FIGURE 19 THE R3-MODEL OF REACH, RANGE AND REALM ............................................... 57
FIGURE 20 GRANULARITY OF SERVICES ............................................................................... 58
FIGURE 21 EXAMPLES OF REALM .......................................................................................... 58
FIGURE 22 DIFFERENT SERVICES OF THE SAME GRANULARITY ......................................... 59
FIGURE 23 R3 OF THE ENTIRE INFRASTRUCTURE .................................................................. 59
FIGURE 24 SAME FUNCTIONALITY, DIFFERENT SERVICE GRANULARITY ............................ 60
FIGURE 25 METHOD OF ANALYSIS ...................................................................................... 62
FIGURE 26 PROPERTIES OF SOA THAT AFFECT ENTERPRISE ARCHITECTURE ..................... 63
FIGURE 27 THE DOMAIN OF RESEARCH QUESTIONS ONE THROUGH SEVEN ....................... 64
FIGURE 28 THE DOMAIN OF RESEARCH QUESTIONS EIGHT THROUGH ELEVEN .................. 65
FIGURE 29 PROFESSIONAL CATEGORIES .............................................................................. 71
FIGURE 30 EXAMPLE OF HBDM CLASSIFICATION (ADAPTED FROM IBM 2006) ..................... 77
Introduction

1 A vision of IT meeting the needs of an evolving business

The aim of this very first chapter is to establish the thesis’s problem domain, in a simple but comprehensive manner. When continuing with the report, many of the themes and much of the terminology introduced in the chapter will be recognized by the reader.

1.1 Business strategy and IT support

Change is the new imperative of business managers. Any company that stops innovating in today's highly competitive business environment will, sooner than later, be overrun by its competitors (Moore, 2006). Continuous incremental change to gain operational efficiency has for the last decades been the prime competitive advantage of many businesses; the company which produced at lowest price automatically won the market. But as the race heads on the lagging group are gaining on the leaders. Today operational efficiency has become a commodity. In response leading companies have revitalized their attempts to compete on market positioning and are now tailoring their whole organizations to suit well specified market segments (Porter, 1996).

“Business process” is a term for taking an arbitrary input and performing a number of activities in order to transform the input to an output with a higher value. Increasing operational efficiency almost always includes removing non-value-adding steps from the process, for instance removing an unnecessary storage point between two machines. The new market centered competition relies on process modifications to an even higher degree as companies aim to create new capabilities by integrating and radically reconfiguring their processes (Stalk et al. 1992).

Information technology supports almost every aspect of a modern company’s operations. IT-systems directly or indirectly supporting business operations, in the thesis referenced as IT-support systems, are vital to the success of the organization. From keeping track of deliveries and inventory to performing financial transactions and making sales calls, almost all company processes are to some extent encoded in the IT-support. It is thus not the technology per se that acts as a differentiator, but the processes that are wired into it. (Broadbent & Kitzis, 2005)

In their efforts to increase operational efficiency many companies in the industrial sector have adopted enterprise systems. An enterprise system is essentially an attempt by a vendor to build one system that governs every activity in the entire company; some prominent suppliers of these systems are SAP and Oracle. The problem with buying an enterprise system is that the buying company commits itself to using the set of processes that is already wired into the system by the supplier, which may put severe limitations on the company’s ability to compete on differentiation (Davenport, 1998). Even when disregarding the paramount task of changing the way people work in an organization, to fit the way the system is working, and recognizing that most systems can be configured using thousands of parameters, the fact remains that the processes encoded into the system are generic and can be bought from the same vendor by any competitor.

Realizing that a differentiation based business strategy requires at least some customization of the IT-support, many companies have chosen to use a number of
different systems that are good for different purposes. The implementation often takes the form of "system silos", i.e. one system that handles all operations for one department, business unit or function. System silos are especially notable in corporations made up of several merged companies that each had a system before the merger.

The approach a company ultimately takes depends on the strategic view it has on information technology. The strategic view reflects IT’s importance in the company, both in terms of the opportunities IT is expected to create and the commitment to IT the business organization is willing to make. (Weill & Broadbent, 1998) For the owner of a hair salon IT is most likely a non-issue, something that should just work and be cheap to buy. An airline executive may, on the other hand, think of IT as enabling new and innovative business practices; discovering previously unknown routes, pushing around spare parts in a revolutionary new way or whatever it may be.

1.2 Challenges related to system integration

Any company with more than one system will inevitable run into the problem of integration. Integration essentially means making several system talk to each other. A simple example from the banking industry may be that of international money transfers. Back in the day, the amount of money the customer had on his or her account was kept in a book and transfers were done by telegraph. In the 1960:s the ledger was computerized and in the mid 1970:s the telegraph was replaced by a computerized inter-bank transfer service (SWIFT). As long as inter-bank transactions were quite infrequent it worked fine for the bank teller to manually make a withdrawal from the account and then type the amount and the destination into the money transfer system. As volumes grew, the manual handling became too personnel consuming and risky; the systems needed to be integrated so that a withdrawal was automatically made when a transfer was initiated.

Conducts such as process integration and the use of several specialized systems create especially high demands on integration. But business restructuring also calls for the decomposition of existing systems. Business process reengineering often entails removing or automating steps in the process, which also means that the steps must be altered in the IT-system that supports the process. Altering an existing system or group of systems is both time consuming and expensive.

One of the main problems of integration is the complexity that comes with it. Complexity is the state of a whole that is made up of complicated parts or where the relationships between the parts are imperfectly known (Merriam-Webster, 2006b). Computer systems are per definition complicated and it doesn’t get any better when they are entangled, especially since it is common for larger corporations to have hundreds of systems with thousands of integration links. Complexity does not necessarily mean trouble. An organization which manages and controls the complexity of its systems has a competitive advantage, since it is hard for competitors to copy the functionality (Anderson et al., 2006).

1.3 Service-Oriented Architecture

One way to tackle some of the complexity related to integration and rebuilding of processes is to divide the IT supports into components. Each component will have a very well defined purpose and can be reused whenever functionality of that kind is needed.
IT-support composed from several components can have some vendor supplied standard components and some custom made components, to accommodate for uniqueness whilst keeping costs down. If the IT-support later needs to be rebuilt, for example due to a change in the business process, individual components can be replaced with new ones. A prerequisite is, however, that the components are not integrated, or coupled, in such way that a change in one component makes another stop functioning; the components need to be loosely-coupled. (Szyperski, 2002)

A Service-Oriented Architecture is essentially a collection of services that communicate with each other. An SOA service can be pictured as an instantiated and running component, providing any user with a well defined piece of functionality. A service may for instance be defined as a “send bill”-service and used in any business process with a need to bill a customer.

The thing that differentiates SOA from other software architecture initiatives is that an SOA is designed to mimic the flow of business processes (Erl, 2005). Essentially SOA prescribes one distinct service in support of each activity in the business process. The “send bill” service may for instance be a part of the IT-support for a business process with the steps “take order”, “check inventory”, “schedule delivery” and “send-bill”. The other steps will be supported by separate services.

The benefit is that the IT-support becomes more flexible to change with the business process. In two years the process may have to look like “take order”, “charge credit card”, “check inventory” and “schedule delivery”. If the process is supported by SOA, three-quarters of the program code directly handling the business transaction – the business logic – may be reused. If, on the other hand, the whole process were wired into a single system, a lot of code might have to be rewritten and if the system were supplied by a vendor the functionality might not be available at all. Of course the above example is a very simplistic one, but it still serves its purpose. The point is that the SOA concept encompasses more than how to build good software – it is an architectural style which attempts to bridge the gap between IT and business. From a business perspective, SOA is not so much a technology issue. It is rather a matter of developing a framework within which business problems can be defined and solutions can be implemented in a coherent and repeatable way, exploiting reusable and loosely coupled services.

For companies deciding to embark on an SOA venture the major question of concern is: What services should we build and which functionality should each cover? Given the example above the reader can now probably think up a number of things that would make good services. But few processes are as simple as the four-step order process and even when they are, the problem of scope is apparent. Should the same “take order” service support both orders over the counter and in the web shop; the need for detailed information may be greatly reduced in a face to face situations. Should the same “send bill” service be used to send bills to customers in all countries; regulatory demands and payment options may vary greatly on such document in different countries.

The breadth of functionality each service encompasses is usually referred to as the service abstraction level, or the service granularity. In general large grained services are preferred in an SOA (Newcomer & Lomow, 2005). The basic assumption of this paper is that the service abstraction level can be precisely described using a set of underlying factors and that the optimal granularity is contingent on an organization’s view of information technology.
1.4 Handelsbanken’s service venture

The study will be carried out at Handelsbanken one of the four large full service banks in Sweden. IT is a very intertwined part of the bank’s value proposition to the customer; almost no customer interaction or business transaction would be possible without adequate IT. The architecture consists of over 500 systems and well over 10,000 applications, serving several million customers and spending about a quarter of the total costs of the bank. Because the IT support has such profound impact on the bank’s operations, almost all the systems have been custom built by the internal IT department.

The bank has for a long time consciously worked in a component fashion, hence the amount of systems. Over time, demands on integration between different parts of the business have grown and many systems today overlap in functionality. Traditionally, integration has been done by programming a specific integration module in both the system that provides the information or service – the service producer – and in the system that uses the information or service – the service consumer. The many customized integration links have created a maintenance problem and the bank have therefore started to think in terms of single interface services, like the ones used in SOA, where many consumers access one general integration module in the producer system.

The size of the system architecture and the relative importance of IT for the business, together with the service concept, make Handelsbanken a prime object for studying the benefits and challenges of an SOA.

The system PARI is the bank’s first major attempt to create a general reusable service. The task of the system is to collect all information about parties involved in transactions with the bank – customers, branch offices, etc. – in one single place and publish a service that can be used by all other systems that need the information. PARI has been in production for a few years and functions very well overall. However, since the system went on-line complaints regarding it have been raised by several actors. Architects in the central architecture department suspect that the problem may be rooted in the granularity of the service.

In addition, the fact that customers are becoming more international and have a stronger preference for alternatives to the traditional branch office channel, such as internet banking, has made coherence in customer meetings a strategic priority of the bank and, as a consequence, the construction of general reusable services has become a new focus area of the IT department. It is therefore vital for the bank’s architects to increase their understanding of service granularity and how it contributes to creating good services.

2 Research objectives

In order to meet customers’ demands in today’s marketplace, corporations must continuously rework their processes for efficiency and innovation. The side effect is an increasing complexity, especially on the boundary between business and IT. It has been proposed that a Service-Oriented Architecture based on coarse-grained services, each supporting an individual activity in a business process, will enable more efficient business process redesign.
2.1 Aim of the investigation
The objective of this thesis is to factorize the notion of service granularity, explicate how the factors are interrelated and explain how an organization’s strategic view on IT affects the optimal service abstraction level in a Service-Oriented Architecture (SOA).

2.2 Scope limitations
In order to make the scope more precise, a number of limitations have been applied. The limitations serve both as a clarification of the above stated objective and to avoid areas and practices that could possibly distort the result.

2.2.1 Focus on IT
The thesis will focus on how to organize good IT-support for business users, not the business itself. It will therefore not evaluate any empirically studied business concepts, business process or business logic. The limitation should serve as a clarification to the reader. The thesis will not enquire into how new business is created by IT, but how the development of business is best supported by IT.

2.2.2 Focus on architecture
The central theme is IT-architecture with an emphasis on how disparate IT business systems interconnect. The thesis will therefore not cover parts of IT that are concerned with areas such as security, storage, office software etc. The limitation should not have any implications on direct findings of the study since the factors which describe service granularity and their relation to the strategic view should be independent of any specific technical matters. For users of the developed model it is, however, vital to have a clear understanding on how their specific technical platform affects the functionality of the architecture.

2.2.3 Focus on service granularity
The thesis will concentrate on service granularity. Other aspects of SOA will only be investigated in cases where there is a direct relationship with granularity or the surrounding complexity. The aim is to make the scope of the study more precise in order to provide a deeper understanding of the specific subject. The reader should be aware that the thesis, because of the limitation, does not work as a general introduction to SOA or the technical concepts and ideas underlying SOA.

2.2.4 Focus on a single case
Because of practical limitations the scope of the study has been reduced to cover the PARI system and systems using party data in Handelsbanken. Whilst a single case study may negatively affect the possibility to generalize the results to cover other organizations, the reader should notice the apparent advantage of the opportunity to study a mature production system. Given that SOA is a very new concept, the possibilities to analyze a large scale system with a key function are rare. The researchers therefore believe that an in-depth analysis of the PARI system will prove more useful than a survey of a number of more lightweight systems. (A further discussion of the researchers’ scientific view and the investigation method employed will follow in chapters 5-7.)
3 Background
The introduction should have provided the reader with a basic understanding of the premises this work is built upon. This chapter will provide a more comprehensive background on the involved units in order to aid the reader’s understanding of the literature review and the empirical research findings.

3.1 Introduction to Handelsbanken
Svenska Handelsbanken is one of Sweden’s four premiere universal banks. Universal banking entails offering the full range of services in the banking area, from retail banking and insurance to corporate banking. The domestic branch coverage is very good, with over 450 branches all over Sweden. Over the past 15 years, the bank has expanded internationally with close to 150 branch offices in the Nordic countries and Great Britain. In addition there are corporate units in an additional 15 countries. (Handelsbanken, 2007a)

In 1992, the bank also started to diversify into the closely related life insurance field, with the acquisition of an insurance company later renamed to Handelsbanken Liv. The bank has since acquired insurance company SPP as well as the mortgage specialist Stadshypotek. In addition to these diversifications, the bank has acquired several Swedish and foreign banks over the years, starting with the acquisition of Bank AB Norra Sverige in 1914. (Handelsbanken, 2007b)

Handelsbanken’s core strategic principle is that the bank has “one goal and two means”. The goal is to accomplish higher profitability than the competitors’ average and the means are to have lower costs and more satisfied customers. The customer satisfaction is created by the decentralized organization, with a high level of intimacy between the local branch office and the individual customer. Profitability and quality is always prioritized over volume. The rate of goal achievement has historically been high, with 34 straight years of profitability above competitors. (Handelsbanken, 2007a)

3.2 IT at Handelsbanken
Information Technology is essential to the running of the bank. Bank tellers rely on IT for every customer meeting, customers conduct their business online, and every product sold requires a supporting IT. The information technology system is as fundamental to the running of the business as the production line at Ford or Toyota, if not even more.

The bank’s IT department was founded in the early 1960:s, long before commercial software packages and enterprise systems. Hence, most of the currently running software is custom made and developed in-house to suit the specific needs of the business organization. There are more than 10 000 currently running applications, divided into some 500 systems. Each system supports a function or a product, such as keeping track of customers’ bank accounts or stock purchases and naturally there is a great need to interconnect these systems. Customers want to be able to use money from their savings account to make stock purchases, but if systems do not “speak” to each other, there is no way to accomplish the transaction that does not entail manually withdrawing money from the savings account and then using the funds to purchase stocks.

Building and maintaining the bank’s IT capabilities uses a lot of personnel resources, of a grand total of approximately 10 000 employees, almost 1 300 of whom work in the IT-
department. There are numerous professions with distinct responsibilities; a few examples are:

- programmer – writes the actual code in new and updated systems
- database administrator – overlooks database activities and helps systems access the database in the most efficient manner
- architect – creates the “building plans” for systems and groups of systems

3.3 Introduction to Service-Oriented Architecture

The name Service-Oriented Architecture tells a few things about the phenomena itself. First, and foremost, we are dealing with architecture. An IT-architecture defines how to design an IT-system but not necessary how to actually realize it, or in other words: “architecture specification is to an organization what an urban plan is to a city” (Erl, 2005, p. 87). In an environment with many specialized systems, the IT-architecture defines how the systems interconnect and together realize a complete IT support for the business. Architecture rules which systems and components exist in the environment and how the information exchange between them takes place.

Similar to city planning, IT-architecture does not have to follow any strict rules; just as each city has an individual plan, each organization has its own IT-architecture. Service-Oriented Architecture is best seen as an architectural style, a set of rules the IT architect can choose to follow when designing systems. SOA is therefore not tied to any particular programming language, vendor, product or standards body (Newcomer & Lomow, 2005, p. 13). It is also important to recognize that SOA services are a strict computer to computer affair; humans do not enter data directly into SOA services, but rather use some kind of interface application which in turn calls one or more SOA services.

The service-oriented part of the name indicates that SOA is about organizing computer systems as services, in contrast to, for example, client-server architecture or distributed Internet architecture (for an analysis of how SOA differs from other architectures see for example: Erl, 2005, p. 88-108). The concept is very simple and borrowed from the real world. Each service takes some kind of input, does something with it according to a set of rules (usually referred to as logic) and produces some kind of output. For a real world example: A bartender (service) takes an order (input), mixes according to a recipe (logic) and returns a drink (output). The bartender encapsulates a very distinct unit of logic – the skill to mix a drink given only its name as input. A hairdresser encapsulates another very distinct unit of logic – the skill to cut hair in a fashionable manner. One would not go to a hairdresser to get a drink or a bartender for a haircut.

SOA services also encapsulate distinct pieces of logic. One service may for instance be responsible for incoming orders, another for customer data and a third for processing credit card payments. The idea is that, just as real world services, SOA services can be combined to form a business process. Another real world example is the food delivery process of a restaurant: A head waiter (service) appoints a table and takes the order, the chef (service) prepares the food and finally a waitress (service) delivers the meal. The three distinct services performed by the head waiter, the chef and the waitress in combination provide the desired result as illustrated in Figure 1. In chapter 9, it is further explained how SOA services are linked to form a business process.
The restaurant example highlights three other key aspects of SOA, namely flexibility, reuse and loose coupling. Imagine that the restaurant up until now only served food, but there is an increasing demand for ordering cocktails as a starter. The owner therefore one afternoon decides to hire a bartender (service) and form a drink delivery process. The process has the exact same steps as the food delivery process, with the exception of the chef being replaced by the bartender. In the new process, two services are reused, the head waiter service and the waitress service and hence the new process can be speedily introduced as seen in Figure 2.

The flexibility to add functionality to the restaurant’s offer comes from the separation of different kinds of logic into services. If the owner herself took the order, cooked the food and delivered it to the table, it would likely take much longer to introduce the drink delivery process, since she would have to wait until she had mastered the skill of mixing drinks. The reuse of services is made possible by the loose coupled nature of the three original services. The waitress does not need to know or care about how the food was prepared to deliver it to the table and can therefore be used to deliver any other item to the table.

The communication in the restaurant is made possible by the exchange of messages. The head waiter notes the order and the table on a piece of paper – the message – and passes it on to the chef, who reads and acts on the order part, before passing it on to the waitress who reads and acts on the table part. Which table is which and what dishes can be ordered are agreed on beforehand and form a common data format for the exchange between the services.

SOA services also communicate using messages. Just as the order in the restaurant SOA messages are complete, in that they contain everything that is needed to complete the transaction (in the example, both the order and the table). SOA services are usually implemented as web services, which mean that messages are contained in an SOAP
message envelope and are transported over http (see Figure 3). SOAP is a standard that allows for addressing messages and passing metadata together with the message. Both the SOAP envelope and the actual message are usually encoded in XML to aid interoperability and ease development. (Erl, 2005)

![Figure 3 SOA message exchange](image)

Services also have the property of being composable. Composition means that one service uses one or more other services to carry out its task. A specific functionality can thus be built by combining a number of generic services into one. Generally, services are ordered into layers depending on their properties as illustrated in Figure 4. As with other architectural issues the number of layers and the properties of each layer may be arbitrarily chosen. The top of the service layer usually contains task centric business services, which purpose it is to compose sets of other services into business functionality, for example, an order entry service. The second layer contains entity centric business services. The services in the second layer still perform a business function but do not generally contain process logic, instead encapsulating a specific entity. One example of such service is a customer register service that stores and retrieves data about customer entities. The final layer is the utility application layer, which contains services without any business function. Examples may be security policy services and services which access legacy systems. (Erl, 2005)
3.4 Handelsbanken’s system for managing Involved Parties

An involved party is generally defined as a person or group participating in a business transaction. In the case of Handelsbanken, this encompasses customers – corporate as well as private – together with the bank’s different branches, corporate units and employees. All parties are collected in one single system called PARI. The system contains all personal information about the registered parties that the bank needs to carry out its business and comply with regulatory demands. Since the bank has so many responsibilities toward its customers and the government, the amount of information that needs to be stored about each party is quite comprehensive. For example, the bank needs to keep track of which tax office customers and employees pay their taxes to, if they are in the state of bankruptcy and, if required by the customer, the address of the customer’s summer house. The party system is, in other words, a lot more than a simple register with names, addresses and phone numbers.

4 Reading guidelines

The purpose of this chapter is to provide the reader with a brief overview of how the report is structured, in order to clarify how the different parts are related and how the arguments will be presented.
4.1 Structure overview

The report is divided into six main parts. The first part, this introduction, contains the problem formulation and the research objectives.

The methodology part which follows describes the researchers’ scientific view, how the subject was approached and how the work has progressed throughout the investigation. These are important issues to understand for anyone who wants to assess the accuracy of the data or use the study in a larger context.

The third part of the report contains a review of the current state of science and presents research findings in areas directly related to the area covered in this report. This part is most vital to fulfilling the objective of this report, since many conclusions can be drawn by distilling and applying research work of others to the problem at hand. The aim of this report is not to invent any radically new theory, but rather to provide a scientific explanation to already encountered phenomena.

The first chapter of the literature review treats the strategy pyramid, which divides company management into strategic, tactic and operative levels. The strategic level deals with long term objectives and questions such as “which business are we in?” and “How should we position ourselves in the eyes of the customer?”. The first point made in this chapter is that businesses today must continuously evolve to stay competitive on the market. The second point is that IT must evolve when the business evolves and that, in some circumstances, business can not evolve unless IT drives the change. The role IT should take in a company must be contingent on the overall company strategy. The strategic view on IT decides which approach a company takes to enterprise wide integration and sharing of resources. The final conclusion is that the inherent complexity of IT can be a serious inhibitor to business development if not managed properly.

The tactics section discusses two different approaches a company can take to translate its strategies into what should actually be done. The general conclusion is that companies wishing to flexibly respond to market demands must make sure that their IT investments is governed by company strategies rather than short sighted business unit concerns. These companies should also proactively work with reducing complexity, or the sought for flexibility will be lost. The last, operational, level in the pyramid is more precisely explained in the chapters following.

In the second chapter in the literature review an architectural approach to manage complexity is proposed. The idea is to manage complexity by isolating functionality in components. A Service-Oriented Architecture is essentially an componentized architecture that mimics business logic and communicate without being hard coupled. In the chapter the ideas of SOA is described and the architectures potential for creating a more flexible IT-support is discussed. One central point is that SOA is more than technology and that the greatest obstacles can not be referred to technical aspects but to business issues. One of the more prominent of these aspects is how to define the service scope, i.e. how much functionality that should be covered by a single service.

Clarity of scope definition is very important considering the two somewhat conflicting objectives the architecture intends to resolve. First an SOA service should correspond to an activity in a business process, in order to enable rearrangement or elimination of services as activities in the supported business process are rearranged or eliminated. The second objective of SOA is to create greater cost efficiency by reuse of services in several business processes. Ideally complete information about processes are held at
design time, in which the problem is reduced to the skill needed to spot process patterns similar enough to be supported by one service. This is, however, seldom or never the case. The third chapter of the literature review is devoted to finding a structured way of approaching the problem of service granularity.

The proposed R³-model defines granularity in the dimensions of reach, range and realm. The three dimensions together captures service granularity as a volume. It is concluded that complexity puts a restraint on how large this volume can be and still be manageable, hence service developers are forced to a trade-off in spatial extent.

In the last chapter of the literature review, chapter 11, the requirements for further research are discussed and a number of specified research questions are formulated. These questions form a basis for the case study and the following analysis.

The fourth major part of the thesis is the investigation. The case study describes various aspects of Handelsbanken’s IT operations relevant to answering the research questions. Due to the extent of the bank’s IT operations a focal point have been chosen for the studies, namely the party system PARI.

After the case study follows the analysis and discussion. Chapters 15 and 16 contain an analysis based directly on literature and empirical findings. The primary aim is to further examine the R³-model and its usefulness. The next chapter concludes the findings of the report and the final part contains a discussion of these findings based on the researchers own opinions and personal understanding of the subject.

4.2 What to read and what not to read

The thesis is structured as a continuous discussion on the topic of service granularity. To get a good comprehension of the material the authors therefore recommend reading it in order from the first page to the last. Time constrained readers may, however, read the report in different order.

If time is short authors recommend to start with reading introduction in chapter 1, the research objectives in chapter 2 and then move straight on the conclusions in chapter 17. To get a perspective on the research findings and their usefulness it is then recommended reading the literature review and the case study. The authors would like to caution against reading the analysis and the discussion before reading the case study and literature review, since the reasoning in both parts will rely on the reader having acquired an understanding of the case.

Non academic readers may also choose to skip the methodology part, since these chapters do not have any immediate bearing on the conclusions, but is rather included as documentation of the work process for quality assurance reasons.
Methodology

5 A systems approach will be used in positivistic manner

This chapter will explain the authors general view on science. The view a researcher take on science affects how the investigation is carried out, what kind of result that is delivered and to what extent the result can be generalized and applied on other situations.

5.1 The main aim of the paper is explanatory

According to Lekvall & Wahlbin (2001) there are four generic methodological orientations of a study:

- Explorative; gives rudimentary knowledge of a certain area
- Descriptive; describes the facts and conditions in a certain area
- Explanatory; explains how different variables and relations are interconnected
- Predictive; predict what will happen; the major connections need to be known to be able to achieve this

The orientation of the study dictates what kind of conclusions that will be drawn and is a separate question of its own. In accordance with Lekvall and Whalbin the thesis will have a mixture of the above orientations. The objective is to factorize the notion of service granularity, a descriptive aim and then explicate how the factors are interrelated, an explanatory purpose, onto explain how an organization’s strategic view on IT affects the optimal service abstraction level, clearly an explanatory objective as well.

5.2 The thesis has a positivistic scientific point of departure

Science is practiced within different conceptual frameworks, i.e. how the researcher and the world the researcher studies are interrelated. There are two rather distinct attitudes, often referred to as paradigms, a positivistic or a hermeneutic paradigm. The difference between the two originates from if the researcher aims to explain or to understand the topic in question. (Arnbor & Bjerke, 1997). This may seem like a matter of phrasing, but it will as argued below, influence the choice of research approach. These paradigms are often presented as antithesis, there is however no need for that, according to Gummesson (2000). This paper will use a positivistic scientific point of departure, although in accordance with Gummesson and Arnbor and Bjerke, elements from the hermeneutic paradigm will be used as discussed below. Let’s first take a brief look at the definitions:

Positivist paradigm, or Explanatics as referred to by Arnbor and Bjerke, considers the reality created by nature, no matter whom and where you are, the perception of reality is the same (Arnbor & Bjerke, 1997). With its roots in natural science, the positivist paradigm is based on the assumption that science is exact, verifiable, and objective (Patel & Davidsson, 1991). They assume that the same methods that have proven their values in analyzing the classic natural sciences are applicable to the material of social sciences as well (Arnbor & Bjerke, 1997). Hermeneutics represents a reaction to the
awkward rigidity of positivism. Instead of trying to explain casual relationships by means of objective facts and statistical analysis, hermeneutics uses a more personal interpretative process to understand reality (Gummesson, 2000).

The main goal of the thesis is to explain how an organization’s strategic view on IT affects the optimal service abstraction level. The generalization and abstraction of theories is one of the most fundamental parts of the positivistic paradigm. In specific, this thesis concentrates on explanatory and descriptive issues, in contrary to interpretation and subjective understanding, also an essential part of the positivistic paradigm.

The researcher of this study will also strive to maintain a clear distinction between facts and value judgments, in search for objectivity, which also is a very central part of the positivistic paradigm. Furthermore, the researchers will distance themselves from the investigation as much as possible i.e. take on the role of an external observer. However, this is not entirely true since the authors are employed by the contractor and the work is performed at the contractor’s location. The researchers will therefore in some aspects take on the role of what Gummesson (2000) define as Action Scientists, which will be explained further below.

5.3 **A system view on method will be employed**

Arbnor and Bjerke (1997) define six other paradigms or categories of knowledge specific to social sciences, which can be seen as degrees of a positivistic or a hermeneutic paradigm as seen in Figure 5. The degrees are defined in four dimensions, however; reality is considered to be objective and rational or as subjective and relative, the relations to philosophy is decreased or increased, if and to what extent explanation or understanding is the lodestar of knowledge, and lastly if results are general and empirical or specific and concrete.

![Figure 5 Research approaches (adapted from Arbnor & Bjerke, 1997, p. 44-46)](image)

Furthermore, Arbnor and Bjerke introduce three methodological approaches, which corresponds to different degrees of a positivistic or a hermeneutic paradigm; the systems
approach, the analytical approach, and the actor approach, as also seen in Figure 5. These are quite different and the choice of methodological approach will therefore have significant impact on how a study is conducted. In this thesis, the systems approach will be used.

Below follows an examination of the different approaches in a bit more detail. The systems approach argues that the whole is more than just the sum of the parts and that the relationships between them add synergy effects to the system. The knowledge is system dependent and the parts are explained by the characteristics of the whole. There is a very close relationship between business processes and IT-systems, and since interaction between IT-systems and business processes is a fundamental and key issue of the research, the systems approach seems to be the most appropriate.

However, it is important to notice that the analytical approach and the system approach overlaps to a great extent and that the chosen approach covers parts of both the positivistic and the hermeneutic paradigm, but mostly the positivistic, which fits the thesis and is in coherence with previous argumentation.

One of the fundamental differences between the two approaches is the content of determining a relation. An analytical approach aims at finding causal relations, that is, cause-and-effect relations, which are both necessary and sufficient to describe any given effect to a cause. The systems approach postulates a more complex reality and recognizes that this hardly is possible in social science. The aim is therefore to describe finality relations, the relation from producer to product. A producer is neither a necessary nor a sufficient cause for a given product. (Arbnor & Bjerke, 1997) This thesis acknowledges that each producer is only one of several possibilities leading to a particular product, the concept of equifinality. In the same manner can each producer generate many alternative products, the concept of multifinality (Arbnor & Bjerke, 1997). As mentioned earlier and will be discussed further down in more detail service granularity is neither the only parameter effecting complexity, nor is complexity the only property affected by granularity.

In accordance with the just stated, the analytical approach is too rigid and simplifies the reality too far since this approach regards the reality as being additive and thereby does not consider the interaction and correlation between different parts.

The actor approach on the other hand proposes that the whole consists of a structure of meanings, which are socially constructed, i.e. are influenced by, and influences the humans. The whole is understood by the insight of the reality of the actors, consequently the knowledge is dependent on the individual, thus subjective. The scientific point of departure is positivistic and is it not possible to combine that with the actor approach, as seen in Figure 5.

5.4 Induction and deduction as different steps of the analytical approach

The aim of research is to gain a deeper understanding of the world, which is done by relating theory and reality. The researchers’ approach to the relation between theory and reality may therefore have a vast impact on the conclusions presented. The two major approaches are deduction and induction. Deductive research primarily tests existing theory, whereas inductive research primarily generates new theory (Gummesson, 2000, p. 64). In a bit more detail; induction depart from empirical data
from which patterns are searched for and models and theories constructed (Björklund and Paulsson, 2003), whereas deduction originates from theories. The theories lay the foundation for predictions which the researcher then tries to verify through collected facts (Björklund and Paulsson, 1997).

Arbnor and Bjerke do not separate these as two different approaches but treat them rather as different steps of the cyclic nature of the analytical approach which both starts and finishes with facts, Figure 6. This thesis will be analytical in the sense that the starting point will be beforehand known facts from which theories will be made by induction. The theories will be further expanded into the unknown by deduction. If a strictly analytical approach would have been chosen these theories or hypotheses would have been tested, verified or falsified. This will not be the case in this study, neither will hypothesis be presented nor will they be tested. Instead the systems approach will be used as a source of ideas from which finality relations can be formulated. Studies for determining finality relations always contain aspects of induction (Arbnor & Bjerke, 1997).

Many authors argue that a third option, to induction or deduction, is to altering between a theoretical and an empirical perspective and interpretation, a so-called abductive approach. This is however not a third approach according to Gummesson (2000) since “...all types of research becomes an iteration between the deductive and inductive”, the important distinction between deductive and inductive is the starting point. The alternation between inductive and deductive is also conceptualized in Figure 6 by the cycle, the facts that end one cycle is the beginning of the next (Arbnor & Bjerke, 1997).

**5.5 Systems approach in conjunction with Action Science**

As will be explained in greater detail further down one of the most important aspects of the systems approach is that the problem is not beforehand stated and no hypothesis can be made. The process of formulating the problem is a learning process itself which includes both literature reviews as empirical studies. The researcher has to be in contact with the system studied in order to formulate finality relations (Arbnor & Bjerke, 1997). This goes hand-in-hand with Gummesson’s reasoning about Action Scientists. The most
important aspects are: access to empirical real-world data, action and continues adjustments. Management is an applied science that has to be connected to real-world data. “Theories concerned with processes in organizations must be generated primarily on the basis of this real data (inductive) and be assisted by logical deductions from established theory.” (Gummesson, 2000) Moreover, action scientists are supposed to take the result to another dimension and both produce general knowledge and applicable knowledge that later can be put into action and create value for the contractor. Continuous adjustments must also be made in accordance with new information and new events and cooperation between the action scientist and the contractor is essential for action science. Gummesson (2000) suggests that this is a natural way of working for the practitioner but somewhat unfamiliar to the scientist.

5.6 Credibility and objectivity

In accordance with the above stated, the researchers have been employed by the contractor, Handelsbanken, and the work has been carried out at the contractor's location, it is of course somewhat influenced by problems and opinions at Handelsbanken. Most of the gained knowledge of Handelsbanken's systems, IT-architecture, and work procedures has been difficult to compare to best-practices in the industry, as the researchers are rather novice within the financial sector.

Still, efforts have been made to remain objective and to relate observations to existing accepted knowledge with a comprehensive literature review. The researchers would furthermore like to point out that no restrictions on how the study was conducted were made by Handelsbanken, and that very little material was classified as too sensitive to be included in the investigation; thus the authors feel that the picture painted by the investigation is as close to the reality as possible.

6 Inquiry approach

The inquiry approach describes more in detail how the scientific view and research approach should be used to fulfill the purpose. According to Lekvall and Wahlbin (2001) there are three main aspects to be concerned when defining inquiry approach:

- Scope of inquiry objects; case study or cross-section study
- Type of analysis method; qualitative or quantitative
- Type of data; primary or secondary

Systems analysis is conducted using traditional data collecting techniques that are adapted to the specific study situation, accordingly to Arbnor and Bjerke (1997). This paper will only address issues within Handelsbanken, which primarily makes this a case study. There will be no effort made to build an economic business case or alike, hence there will be a very limited use of quantitative analysis. Finally, this thesis will have its base in primary data retrieved from interviews. In order to get a comprehensive view, the primary data will be augmented by secondary data in forms of strategy documents, manuals and technical descriptions.
6.1 Qualitative data from case study interviews

The case study approach can, according to Eisenhardt (1989), be said to be a research strategy which concentrates on comprehending the dynamics that exists within single settings, which is in coherence with the system approach and in fact Arbnor and Bjerke (1994) points out that the case study form is a very usual data collection technique. A case study can consist of one or several cases as well as different levels of analysis, i.e. corporate or business level. Cross-section refers to the fact that a cross-section of a target population at a certain time is studied, and conclusions on the whole target population are the aim (Lekvall & Wahlbin, 2001).

The second aspect, type of analysis method, is often divided into two categories; qualitative and quantitative studies. How to define these categories can be tricky and theorists have different standing points in this question: Lekwall and Wahlbin (2001) assert that the separation is made according to how data are expressed and analyzed, whilst Jacobsen’s (2002) definition reads that quantitative studies seek to describe phenomena whereas qualitative studies aim to interpret and understand.

This thesis uses a single case study on three levels; strategic, tactical and operational, as its basis for empirical data, collected primarily from personal interviews. Most of the data collected, but not all, will be of non-numerical nature in order to get a deep understanding of Handelsbanken and the specific topic, which makes the investigation qualitative both accordingly to Lekwall and Wahlbin and Jacobsen.

6.2 Both primary and secondary data was used

Information is divided into two main categories, primary- and secondary information. Primary information is first-hand information, while secondary information is collected by an intermediate part (not by oneself). (Lekvall & Wahlbin, 2001)

The investigation will use both of these categories. As previously mentioned, it will be necessary for the researchers to have personal interaction with business professionals in order to better understand the problem and its context at Handelsbanken. Personal interviews are perhaps one of the more time-consuming methods of data collection but it also enhances the process flexibility in alignment with the system approach and action science. The primary data collected in the interviews and the secondary data will be presented together in chapter 12-14 in the form of a rich case study.

In order to develop a pre-understanding to interviews or in depth knowledge of mainly specific systems and technologies at Handelsbanken, secondary information in forms of manuals and technical descriptions will be used. Secondary information such as strategy documents and alike will also be used in data triangulation, in order to increase the reliability, during interviews as well as afterwards in an analysis phase.

7 The working process

In order to plan the study and choose the adequate method and research design the topic of methodological orientations in the beginning of the methodology part must be further developed. Arbnor and Bjerke (1997) states five different goals, or objectives, of the systems approach: to determine the type of system, to describe, to determine relations, to forecast, and to guide. The similarities to the four generic methodological orientations presented by Lekvall and Wahlbin are noticeable, but a complication is added by the
system approach. From a methodological point of view, determining the type, describing and determining relations cannot be completely separated from one another. This is because the systems approach starts from the presumption of a reality that is constructed of various totalities, each of which contains numerous connections. (Arbnor & Bjerke, 1997) Therefore the five levels of objectives in the systems approach cannot be assessed in a hierarchical way, as in an analytical approach.

Figure 7 Research framework (adapted from Lekvall & Wahlbin, 2001, p. 183)

The thesis will use a framework presented by Lekvall and Wahlbin (see Figure 7) for the process of conducting marketing research. The framework is used to plan a study, and to illustrate the different steps the researcher must take. This framework is however mainly used for the analytical approach. Arbnor and Bjerke also present a conceptual model of the process, seen in Figure 8, in the case of an analytical approach. Once again the similarities are evident, though the U shape of the Lekvall and Wahlbin model accentuates different levels and that the activity on one side corresponds to the activity on the other on the same level. The model has to be expanded since the thesis has a systems approach. It is however a minor one, and in fact not very different from the practical implementations of the model in an analytical study.

Figure 8 Analytical approach (adapted from Arbnor & Bjerke, 1997, p. 292)
Formulating a problem for systems approach is more extensive than for the analytical approach (Arnbor & Bjerke, 1997). Furthermore, to determine relations implies ongoing interactive contact with representatives from the reality being studied in order to clarify increasingly the finality relations that exists (Arnbor & Bjerke, 1997). Therefore there has to be a number of feedback mechanisms among the various stages in the model, see Figure 9. The research continues until the finality control breaks out of these feedback loops, which means that the discovered producers can provide the discovered products. The reader should be aware of that the researcher’s personal judgment plays a much more decisive role in this part than in the case of an analytical approach, and remember that the finality relations are neither sufficient nor necessary.

![Revised research framework](image)

**Figure 9 Revised research framework**

It has been stressed so far that the systems approach cannot formulate hypothesis and that the problem itself cannot be clearly, specific and definitely stated before the study starts. However, this does not mean that the researcher has no clue what to do or how to do it, Figure 10 shows the rough layout of the process. Using systems analysis means to reproduce the pattern and to determine the finality relations of the real system. This is equivalent to describing and explaining in the analytical approach presented earlier.
There exists different variants within the systems approach, and one that should be mentioned in the context of the execution of the study is to what extent the researcher participates. Arbonor and Bjerke suggests one variant were the researcher only does the first three steps of Figure 10, hence no implementation. They call this variant the expert variant and it is the one used in this study, mainly because of the nature of the master thesis and the context in which the study was made i.e. as an academic work. Nevertheless, the reader should keep some obvious risks in mind due to the choice of variant. One risk is that the researcher is not capable of catching the real essence of the system studied or lack of engagement from either the researcher or the system studied. Another one is communication problems at the delivery of the finished proposal.

7.1 Formulating the problem

The systems approach does not permit the researcher to state the problem at the beginning of the study, neither is it possible to formulate hypothesis, as laid out earlier. Instead, the phase of formulating the problem includes data gathering and formulation of new possible finality relations, in a feedback loop as previously discussed, see Figure 9 (for a more detailed discussion see Arbonor and Bjerke (1997) page 296). The figure does not catch the need for literature review in order to formulate new possible finality relations, therefore the extended U-model will be used.

The literature and the early empirical results indicated that the degree of complexity was a major determining factor, both for realizing a healthy SOA and for determining service granularity and the effects thereof. But also, in order to create any sustaining competitive advantage the starting point had to be the strategic intent of the company and to explain how an organization’s strategic view on IT affects the optimal service abstraction level.

Obvious was also that complexity was present at different levels within both the business realm as that of IT. Therefore, how to translate business strategy to concrete guidelines on an operational level, both for business and IT had to be a main theme of the thesis. After another loop the literature with support of the empirical results suggested that complexity revealed itself in different dimensions, which lead to the
development of the R³-model. The findings this far gave the specified research questions presented in chapter 11 in order to investigate the finality relations and thus accomplish the objective stated in chapter 2.

7.2 Data gathering

There are different techniques in order to perform a case. As presented earlier this thesis is primarily grounded on data collected from interviews. There are three major kinds of interviews, the structured, the semi-structured and the unstructured. (Bryman & Bell, 2005). The structured interview is mainly used for quantitative research and aims at revealing objective facts, a much used technique in the analysis approach. If the emphasis on the other hand is on how the interviewee understands and comprehends course of events the semi-structured interview is better suited. The unstructured interview is more like an ordinary conversation and aims at exploring what the interviewee thinks of a certain topic without guidance. (Bryman & Bell, 2005)

This thesis used semi-structured interviews in order to guide the interview but also maintain the flexibility and ability to discover new and interesting leads (Kvale, 1997) in alignment with the systems approach of the thesis. The semi-structured approach also opens the possibility to re-interview some interviewees in order to follow up interesting leads and clarifying complex relations and details, which was done in some cases, also in consistency with the systems approach. The interview followed an interview guide with on beforehand made questions and topics. The questions were necessarily not asked in the same order and all questions were not asked each interviewee, depending on their professional situation.

During the interview protocols were taken by two independent persons and afterwards combined into one, which in turn was reviewed by an autonomous bank employee in order to cross check facts. This was done with the purpose of increasing the reliability of the thesis.

The interviews and the protocols were solely done in Swedish, the native language of both the researcher and the interviewees. The text, including quotations, was freely translated into English in the phase of writing this thesis as suggested by Kvale (1997), in order to spare the reader from a taunting experience by translating word for word what the interviewee actually said. The emphasis has instead been on the intention and meaning.

7.3 Analysis and Conclusions

Most interviews were done within a condensed time frame, however some where done on beforehand and some afterwards. After most interviews were completed a rough analysis of the data was done and the data was arranged into first hand results which are presented in the investigation part. During this work data was screened and irrelevant material was sifted out.

The first hand results were analyzed, the R³-model was verified and the interrelation of the factors of service granularity was explicated. Furthermore, the service granularity’s affect on complexity as well as other quality aspects of Enterprise Architecture were explained, and since SOA is suggested as an approach to Enterprise Architecture other aspects of SOA affecting complexity was also examined.
Handelsbanken’s strategic view on IT was analyzed in order to determine if it was aligned with the incentives of PARI, especially regarding the service abstraction level. The linkage between the business strategies and IT operations was also studied in order to determine the context of the PARI incentives. Finally conclusions were made and a discussion part follows in which the researchers’ reflections and recommendations are presented.
Literature review

8 Three levels of objectives

For corporations to function there need to exist a common understanding of what the organization should accomplish; a goal to reach. The goals – or objectives – have a different precision depending on which managerial level in the organization they are formulated at. It is usual to talk about strategic, tactic and operational objectives as illustrated in Figure 11. In short, this chapter will explain how decisions on business strategy level translate into operational level objectives of information technology.

8.1 Business perspective on strategic level

For millions of enterprises worldwide the rules of business are rapidly changing. In an increasingly higher pace customers expect companies to deliver better quality and more innovative offerings at lower prices than ever. In this section it will be concluded that the new competitive climate requires companies to continuously evolve and refine their business processes.

8.1.1 Business strategy has become concerned with processes

The Merriam-Webster dictionary defines the word strategy as “the art of devising or employing plans or stratagems toward a goal” (Merriam-Webster, 2006a) and places its’ origin back to ancient Greece, where a board of 10 strategi commanded Athen’s great armies (Encyclopedia Britannica, 2006). Norman and Ramirez (1993) apply the word to the domain of business and states that “Strategy is the art of creating value.” (p. 65).

Regardless of the exact definition or domain, strategy is often referred to as a deliberate and universal plan; something that the top executives craft and the organization carries out. As we all know, however, plans can fail and when they do fail they need revision. In the late seventies Mintzberg (1978) argued that strategy corrections are not so much failures as an inherent part of strategy itself. In his article patterns in strategy
Mintzberg (1978) defines strategy as “a pattern in a stream of decisions” (p. 934). In his view plans and stratagems are merely proposed intentions of change. The actual outcome is decided in an interplay between the forces of a dynamic, irregularly evolving environment and a rigid organizational bureaucracy, with top executives as mediators between the two. The environment will demand change, but any plan devised to deviate from the path currently followed by the organization will immediately be stalled by deep-rooted procedures, reluctant stakeholders and, as shall later be conclude in this paper, inflexible IT-systems.

The suggestion that strategic plans must continuously evolve poses a dilemma to business leaders. If it is not good enough to just let the organization run with what it knows; which parts of business should be thrown out and which parts should remain and be cultivated? A lot of thought has gone into solving this problem over the years and several ways to structure the dilemma has been proposed. One of the most prominent is the notion of core competence introduced by Prahalad and Hamel (1990). The idea is to move away from a product centric view of the business and find the common universal denominators – core competences – that serve as foundation for all company products and product lines. The core competences are knowledge about a manufacturing method, a throughout understanding of a complex technology or anything such that provides a core to build product lines around. Generally a core competence should be difficult to imitate for competitors, make a significant contribution to perceived product value and give access to a wide variety of markets. Likely candidates are usually competences that merge and harmonize a number of resources and capabilities into a working system. Prahalad and Hamel (1990) therefore put forward that what must be done to keep up with the evolving environment is to identify the core competence of the company, develop it appropriately and use it as a platform to build future product strategies around.

The core competences, so to say, represent the organizations accumulated and globally unique knowledge on a number of specific topics that can be combined to develop new innovative product offerings (Prahalad & Hamel, 1990). But the knowledge to build new product offerings is not all that is needed to gain dominance in the marketplace, the game is also about time. Time, as in time to production, time to market and time to customer, has become a key source of differentiation in recent years and many company fortunes have been built around managing processes in a skillful way to reduce time consumption. Stalk et al. (1992) uses the word capability to describe how these companies compete.

The capability is much like the core competence’s merry cousin. They both dissociate themselves from the view of products and markets as the fundamental building blocks of strategy and they both advocate a cross disciplinary, rather than a SBU centric, view of the company, but as core competence practitioners emphasizes harmonization and combination of resources and knowledge capability practitioners emphasizes processes and workflow. The cornerstone of the capability view is that, as markets mature and products get commoditized, the competitive focus shifts from the product to the delivery system. (Stalk et al. 1992)

A capability can be defined as a set of integrated, coordinated and jointly optimized processes which together delivers a value sought by customers and hence drives market demand. A company that correctly assesses customer’s needs and builds capabilities to match them will outshine its competitors in that market. If, for example, a company has a more efficient idea-to-market capability it can incorporate new technology faster in its
products and always release news before anyone else, if they have a more effective market-intelligence-to-product process they will be more sensitive to customer expectations, and if the firm manages its supplier-to-customer processes better it will pay less penalty for carrying inventory and be able to keep prices down. (Stalk et al. 1992)

According to Stalk et al. (1992) capability driven companies outperform rivals along five dimensions: First the speed with which they respond to changes in market demands and new technology. Secondly the consistency which they deliver their offers with. The third dimension is acuity, in the sense of clearly seeing the competitive environment and thus be able to anticipate and respond to changing customer preferences. The fourth is the ability to concurrently adapt to several different business environments, here referenced as agility, and at last, innovativeness or the ability to come up with new ways to generate customer value. The key is to identify the company’s most crucial business processes, manage them centrally and invest in them for a long term payback. The authors specifically cautions against conventional ROI analysis when it comes to capability driven competition, since standardization and outsourcing often will be cheaper but will not provide any competitive advantage for the future.

8.1.2 Competing on differentiated business processes

Capabilities, in Stalk et al.’s sense of the word, are built around processes and seem especially applicable in dealing with rather mature industries, where most of the market territory has already been claimed. Since this thesis ultimately is concerned with creating value in the banking industry, which by all means must be seen as a mature market, a further investigation seems justified. Johansson et al. (1993) offers the following definition of the word process:

“A process is a set of linked activities that take an input and transform it to create an output.” (p. 57)

They continue by stating that: “Ideally, the transformation that occurs in the process should add value to the input and create an output that is more useful and effective to the recipient either upstream or downstream.” (p. 57), implying that not all processes do good.

The creation of good processes is the major goal of business process reengineering (BPR). To do BPR is essentially to identify all activities in a process, question which value they add, remove those activities that are deemed non value adding and then optimize, simplify and coordinate the activities that are left. IT plays a crucial role in any BPR effort, since many tasks can be automated (or has already been so, even though the process has not been updated). A key insight is that IT must not be used to automate existing processes but to build new ones. (Hammer, 1990; Johansson et al. 1993)

Johansson et al. (1993) categorizes three motives for engaging in a BPR effort:

1. Cost improvement; BPR for cost reduction will not bring in new customers, but can be a serious help to improve bottom line results if applied to supportive processes. A BPR effort is usually a more radical and speedy way for cutting costs than for example the continuous improvements associated with a lean manufacturing program.
2. Business parity or best in class; For business core processes companies should aim to achieve a best-in-class status. Being among the best in the marketplace is surely good and attracts customers, but it does not mean a definitive advantage since there may be many companies that are roughly equally good.

3. Competitive dominance; To attain dominance in the marketplace the company will have to rewrite the rules of the game and redefine the meaning of best-in-class for the given market. This is achieved through what the authors call BreakPoint Business Process Reengineering.

Breakpoint BPR can be thought of as a tool for firms that wish to become capability centric. Again the focus of attention is the set of processes that drives customer satisfaction and demand; the processes that yield the kind of value that customers are willing to pay for. To know exactly what this means of course needs a great deal of insight into the business, company and customer segment in question, but also a great deal of foresight since customer demands are constantly evolving. (Johansson et. al 1993).

Evolving customer demands continuously raises the barriers of entry, or must do’s, to the market. In the seventies companies could successfully compete on their respective products characteristics, a decade later the barriers had risen and customers also demanded that the product were to be highly functional and inexpensive. In the nineties high quality and innovation was added to the list of required prerequisites and today time-to-market and service are the basis of competition. These are the fundamental requirements that customers place on any company that wishes to be an actor in the market, if they are not fulfilled the company will seriously lag behind. (Johansson et al., 1993)

Johansson et al. (1993), however, argues that no matter how important the must do’s are, they should not be the focus of company attention. If the company is already in the market and intends to stay the must do’s should already have been sorted out, and if it is aspiring to become a player, merely meeting up to the most fundamental requirements will not do. Instead the company should strive to identify the next set of must do’s and get a head start of competitors implementing them in current operations. This set of value-adders are the can do’s. The can do’s represent value that the customer does not yet know to ask for, either because they do not know that it exists or they do not believe that it can be realized.

In 1970 the only company in the world that strived to keep end user prices down by eliminating waste in production using just-in-time principles was Toyota, then mainly operating on the domestic Japanese market. Today Toyota is the world’s second largest manufacturer and auto buyers have become so accustomed to low-price, high-standard vehicles that no company in the industry can afford not to use just-in-time. (Johansson et al., 1993, p. 10-11) By creating, what was then, a BreakPoint manufacturing process and introduce added value (same or better quality to a lower price) as a can do long before any competitors Toyota has been able to catch up and surpass most of it’s competitors and their invention is now a must do in the industry.

The view that it is not enough to just streamline processes for operational efficiency is further promoted by Porter (1996). In his article What is strategy he introduces the concept of a productivity frontier. The frontier represents the state of collected best practices in the industry, in terms of cost position and delivered value. Companies far
behind the frontier can streamline their business and achieve both extended customer value and lower costs at the same time. This is comparable with Johanssons et al.'s (1993) second motive (business parity or best in class) to engage in BPR. According to Porter (1996) this pole position in the market race, however attractive at first glance, is a dangerous one if not properly managed. When a company approaches the productivity frontier trade offs between quality and cost will be necessary. A company used to continually be able to deliver better products or services to a lesser cost will suddenly find itself in a spot where there is not much more that can be done and where a sacrifice of either quality or cost is necessary to keep moving forward. Furthermore, if several companies in the same industry follow the same path toward the productivity frontier the risk of hypercompetition and price war is impending. Johansson et al.'s (1993) solution is essentially to move the productivity frontier by engineering a BreakPoint, but Porter (1996) cautions against relying on this kind of strategy. The diffusion rate for productivity and process innovations is too high. As soon as the productivity frontier is moved competitors will start to copy the method and, sooner than later, catch up in productivity.

In Porters (1996) eyes strategy must be superior to and disassociated from any operational effectiveness efforts. This is not to say that operational effectiveness is not important, but rather that it is so fundamental that it cannot be relied on to create competitiveness in the market. The whole concept of operational effectiveness is so to say a “must do”. Instead Porter argues that sustainable competitive advantage can only be achieved by selecting a unique and defendable strategic position in the market. For example Bessmer Trust Company has tailored all their operations to meet demands from families with a minimum of $5 million of investable assets, which gives them a distinct advantage over banks with a more broad clientele (p. 66). The article offers several ideas on how to slice-and-dice the market to find a position to occupy, but that is beyond the scope of this thesis. The crucial point of the example, however, is that Bessmer has chosen to perform their business in a radically different way than other banks. Their way may not be more efficient, but since the whole business model is tailored to their target customer group (they for instance account for racing horses and conduct customer meetings at clients’ yachts) they can charge premium fees for their service. Porter (1996) states:

"Activities, then, are the basic units of competitive advantage. Over all advantage or disadvantage results from all company’s activities, not only a few ... strategic positioning means performing different activities from rivals’ or performing similar activities in different ways." (p. 66)

The first part of the quote is worth some additional consideration. In order To successfully occupy a selected niche of the market it is not enough to select just a few activities or processes to modify, instead a holistic view with a focus on the workings and

---

1 We want to make the reader aware of that Porter’s definition of the word “activity” differs from the way the word is used in other parts of this paper. In our analysis an activity will be defined, in conformity with Johansson’s (1993, p 57) definition, as a distinctive part of a process. Porter, on the other hand, does not relate activities directly to processes and defines activities as a very broad unit of analysis. Example activities, collected from the Vanguard activity map in Porter (1996), are “In-house management for standard funds”, “On-line information access”, “Limited international funds due to volatility and high cost”, etc.
coordination of the entire company is required. These basic ideas ties Porter’s sense of strategy to Stalk et al.’s capability view of the company, but also aligns it with Johansson et al.’s BreakPoint BPR. Large-scale continuous companywide restructuring will not call for reengineering only of a single process, but a whole lot of them.

8.1.3 Competing on standard processes

So far it has been established that competitiveness in the market is based on differentiation in selection and implementation of processes. Resorting to “best practice” and dumb efficiency optimization when it comes to core process is fallacious and threatens to undermine the company’s position in the market.

On the other hand, several authors argue that companies have a lot to gain if they standardize and commoditize their processes (see for example Hammer, 1999 and Davenport, 2005). The argument for standardization (within the company) is built on the following perceived benefits (Hammer, 1999):

1. A standard process renders less overhead. If the same process is deployed everywhere it will only need one process owner, one IT-support system, one staff training routine and so on.

2. Using the same process in all branches and offices ensures consistency towards customers and suppliers.

3. Uniform processes will also make the company more flexible, since personnel and equipment can be transferred between units as needed in response to shifts in demand.

Standardization of processes within a corporation does not, in any way, interfere with the idea of corporation unique processes. Intuitively it can be seen that many companies, which differentiate themselves from competition using a unique process configuration, enforces a single and mandatory processes for all branches worldwide (think of McDonald’s, IKEA or Ryan Air). The dispute is rather about whether or not industry global processes is favorable for companies.

Stalk et al. (1992) cautioned against letting short term ROI-calculations deceive decision makers into standardize and outsource company infrastructure. Davenport (2005) argues that process standardization among competitors makes sense from two perspectives. First, processes standardization makes benchmarking and quality control possible, this will make it easier to reassure customers that the product will meet requested standards and also work as a motivator for process improvements within the organization. Second, process standardization is a prerequisite for business process outsourcing (BPO) and shared services centers, in which competing companies cooperate in producing a service common to both companies.

BPO and shared services centers has its’ roots in the core competence school of thought. Davenport (2005) writes: “Process standardization may also mean that it’s feasible to combine certain processes with competitors; if these processes offer no competitive advantage, why not?” (p. 107), implying that if a process or task is not essential to “core” it will not matter if it is performed in exactly the same way as competitors. Another reason in strong support for process standardization is the cost and efficiency benefits that can be gained from adopting a standard software package (Davenport, 1998).
8.1.4 Core process should be unique, context processes should be standardized

The originally stated dilemma of which parts of the business to keep and which part to get rid of by now seems to have boiled down to the question of which processes to make unique and which to standardize. Moore (2005a) offers an interesting conceptual solution to our problem in his book *Dealing With Darwin*. He is primarily interested in innovations – including both product and process innovations – and how a company should allocate resources over the innovations life cycle. The thesis is that companies can never stop to innovate and evolve if they want to stay in business.

“What Darwinism says is either evolve or get marginalized. Darwin doesn’t care. At the end of the day you perform or you don’t.” (Moore, 2006)

The way to “evolve” and “perform” is according to Moore (2005a) to move resources from already established innovations to novel ones. Just like Porter, Moore recognizes that any innovation, no matter whether it’s a product or a process innovation, will sooner or later become known and copied by competitors. To aid the analysis the author defines a taxonomy where any given invention can be either mission-critical or non-mission-critical and core or context. Core innovations are those which are novel to the market and offer a high degree of differentiation. Context innovations – again products or processes – are those which are well known and understood by all players in the market. An innovation is considered mission critical when it constitutes a so large part of a company’s revenues that it can not do without it. (Moore, 2005a; Moore, 2006)

Every innovation follows a simple life cycle. First it is invented and if it is a good invention it will offer a potentially high degree of competitive advantage, thus it is initially considered as non-mission-critical core. When it is later deployed it will soon become mission-critical, since customers will want to benefit from the new good that the invention brings. As time passes by the diffusion process will ensure that competitors start to offer the same good in their offerings and the degree of differentiation the innovation brings is reduced; the innovation is degraded to context. Finally when other new and novel innovations enter the market the original innovations relative importance to the company will diminish and it will be reduced to non-mission-critical. (Moore, 2005a)

Competitiveness is built by using revenues from context innovations to fund development and core innovations. Any process innovations that are considered core should be heavily funded and customized to exploit the opportunity for differentiation it offers. As soon as the process becomes industry standard, however, this focus must shift towards standardizing, commodizing and reducing costs. A context process, even if mission critical, per definition will not provide opportunity to charge a premium prize. The challenge with context processes is primarily to get the work done as inexpensively as possible with maintained quality, so that customers are not lost. (Moore, 2005a)

8.2 IT perspective on strategic level

Today, the business significance of information and technology capabilities and assets is higher than ever. IT is something virtually every executive task depends on in some form. (Broadbent & Kitzis, 2005)
The IT strategy is not an isolated phenomenon. The IT strategy has to be in alignment with and derive from the business vision. The following chapter explains the context of IT on a strategic level and what implications on the overall business IT has. It is important to stress that an organization’s IT will never be, and should never be, the sole purpose of the business. Hence, IT has the power of enabling or restraining great business ideas, but will not create any competitive success unless accompanied by relevant changes in organizational routines.

What matters is how business and IT executives, managers and professionals, work together at all levels in the organization. It’s about the vision, motivation, goodwill and persistence of people supported by the effectiveness of relationships and process capabilities around integrating business and technology developments. The technology and systems are not long-term differentiators. (Broadbent & Kitzis, 2005)

8.2.1 Definition of IT-infrastructure

IT has profoundly changed the nature of how business is conducted nowadays; it has increased productivity and contributed to overall economic growth (Lyons 2005). However, the technology in itself, the IT-infrastructure, is best seen as the latest in a series of broadly adopted technologies that have reshaped industry over the past two centuries - from the steam engine and the railroad to the telephone, according to Carr (2003). He argues that the technology’s potential for differentiating one company from the pack, its strategic potential, inexorably declines as it becomes accessible and affordable to all. Infrastructure is simply costs of doing business that must be paid, no lasting competitive advantage arise from infrastructure technologies (Carr 2003). This thesis does not share Carr’s view on IT-infrastructure, limited to just the transportation of bits and bytes. It is therefore vital to define what IT-infrastructure is.

The thesis uses a definition proposed by Weill and Broadbent (1998) which origins from a much broader approach presented in Figure 12. At the base are the technology components, such as computers, databases and operation systems, commodities readily available in the marketplace. The second layer comprises a set of shared information technology services. The technology components are converted into useful shared services by a human information technology infrastructure composed of knowledge, skills standards and experience. This human infrastructure binds the technology components into shared IT services. The infrastructure services in an organization often include computers and mainframes, the management of shared customer databases, and even IT-infrastructure related research and development, aimed at identifying the usefulness of emerging technologies to the business. At the top of the IT-infrastructure, is a set of shared and standard infrastructure applications used by all business units, including accounting, human resource management and budgeting. (Weill and Broadbent, 1998)
The common denominator of infrastructure services and applications is that they are usually quite stable over time and improvements take place gradually. In contrast is the information technology required for business processes, which changes frequently to better suit customer needs or in response to competitor activity. Furthermore, in building new business applications an organization will use the IT-infrastructure services in place in order to significantly reduce time and cost of the new application. (Weill and Broadbent, 1998)

Another puzzling confusion on the subject of IT-infrastructure, or actually on infrastructure in general, is the debate over flexibility verses standardizing of infrastructure. What is meant by a flexible infrastructure and is it something desired? It depends on the scope of the definition of infrastructure. For example: Is the railroad system infrastructure or is just the railroad track the actual infrastructure? In order to achieve a flexible railroad system, properties such as the distance between the rails of the railroad has to be standardized, just like the protocols of the internet has to be standardized in order to achieve a flexible internet. The more standardized bottom line infrastructure is, the more vastly different goals can be accomplished by the infrastructure as a whole, thus more flexible.

**8.2.2 Competitive advantage of IT-infrastructure**

The IT-infrastructure can be seen as, in terms of Prahalad and Hamel (1990), a core competence or as supportive foundation of a capability in terms of Stalk et al. (1992), as
it is created through the fusion of technology and human assets. It is difficult for competitors to imitate, is cross disciplinary rather than SBU-centric, and delivers agility to respond to changes in the marketplace. Weill and Broadbent (1998) also emphasize that organizations with more infrastructure have faster times to market and more sales from new products. Building an infrastructure tailored to an organization's strategic context takes considerable time and expertise. Although the components are commodities, the management processes to implement the best mix of infrastructure capabilities to suit a specific organization are much more scarce resource (Weill and Broadbent, 1998). Further articulated by Seely Brown and Hagel (2003) in their response to Carr: “The underlying technology components may be widely and cheaply available, but the skills required to organize them into high-value architectures are still in very short supply...”.

It will always be possible for competitors to find a way of copying or improving upon any new IT-system. At the end of the day, the only way to add value to a company is from business operational change used in conjunction with IT. In most situations lasting advantage comes only from a commitment to continuous change using IT. (Luftmann, 1996, p. 5)

Furthermore, as noted earlier, business processes will evolve over time, so it is a necessity for IT-systems to be flexible enough to support or even drive that transformation. As Johansson et al. (1993, p. 119) points out in their discussion about BreakPoints “information systems must be utilized to their fullest. In some instances, the BreakPoint cannot be totally achieved without breakthrough use of information systems”. It is however important to remember that as IT-systems become more and more intertwined in business processes and have increasingly higher complexity they will not only become a key enabler of change, but also place limits on what change is possible and how quickly it may occur (Lyons, 2005). In conclusion the competitive advantage of an IT-system derives from the strategic context and the business processes executed by the components.

8.2.3 Organizations have different views on IT-infrastructure

Weill and Broadbent (1998) have in their research identified four different approaches to IT-infrastructure, depending on an organizations view on cost justification, benefits and up-front investments. In moving from none to enabling view the level and extent of top management attention to IT-infrastructure increases, as does the investments and capabilities. However, none of these views is superior across all organizations. A particular view is best suited to each company for a particular period of time. Weill and Broadbent (1998) define the four views as:

A none view of infrastructure implies that the company has no enterprise-wide IT-infrastructure. The organization usually has independent business units with few synergies, which are encourage to invest independently in IT as in other assets. In taking this view the organization foregoes any potential economics of scale or business synergies from the sharing of IT-infrastructure. (Weill and Broadbent, 1998, p. 97)

A utility view of infrastructure implies that expenditures on IT-infrastructure are made primarily to achieve cost savings through economics of scale. The infrastructure is not a strategic resource, but rather a utility that incurs administrative expenses and is a necessary unavoidable service. (Weill and Broadbent, 1998, p. 97)
A dependent view of infrastructure implies that the infrastructure investments are primarily for and in response to well-articulated business strategies. Dependent infrastructure investments are derived from business plans that specify strategic objectives, such as strategic intent and business goals. For example, a bank might invest heavily by consolidating previously independent databases into an integrated customer database. This infrastructure investment is dependent on a current strategy of differentiating customer service through relationship banking. Previously the bank could not determine its full relationship with a customer, as the different account information was spread over a series of databases and data files that had been developed at different times and linked to different account types. (Weill and Broadbent, 1998, p. 99)

An enabling view of infrastructure implies that infrastructure capability is a core competence and strongly integrated with the strategic context. The infrastructure provides future options for implementing current strategies consistent with the future strategic intent. The flexibility of the infrastructure permits a number of as-yet-unspecified business strategies to be implemented more rapidly than in organizations with a dependent or utility view on IT-infrastructure. To take an enabling view, senior management must perceive a flexible infrastructure as a true asset of the organization which provides a competitive advantage and thus over-invest in infrastructure based on current needs. (Weill and Broadbent, 1998, p. 100-101)

In any organization over a period of several years, it is likely that different projects will be undertaken that will cover the full range of views. However, one view tends to dominate the information technology portfolio, according to Weill and Broadbent (1998). Remember that infrastructure capability is a strategic choice, and no single view suits all organizations, but different views have different impacts on a company’s performance.

8.2.4 Two roads to competitive advantage leveraged by IT

The possibility for IT-infrastructure to be a business enabler depends on how IT-infrastructure is leveraged, the strategic context and the organization’s view on IT-infrastructure. Venkatraman (1994) presents a framework for linking the strategy and the role of IT to the possible business impact and the roads ahead. His starting position is that the business environment calls for a strategy based on three intertwined elements: low cost, high quality, and fast and flexible response to customer needs. No one element is sufficient for competitive success.
The framework illustrated in figure 13 has two dimensions; the range of IT’s potential benefits and the degree of organizational transformation. The underlying theory is that IT deployment, superimposed on existing organizational conditions such as strategies, processes and culture, will only generate marginal payback (Venkatraman, 1994). Thus, the benefits accrue in those cases where investments in IT functionality accompany corresponding changes in organizational characteristics (Venkatraman, 1994). The differentiation is not in IT itself but in the new practices it enables (Seely Brown & Hagel, 2003).

It is important to remember that the levels of the framework described below are not conceptualized as stages of evolution because effective strategies do not, and should not, follow any one prescribed model. Higher levels of business transformation entails higher degree of changes in organizational routines, albeit indicate potentially greater benefits. Once again, the emphasis is on what the organization does and not on a certain technology. (Venkatraman, 1994)

The first level of the framework – labeled rationalization – has evolutionary characteristics and requires less change to the business processes relative to the higher levels, which has revolutionary characteristics. The level includes both localized utilization within business units and internal integration between business units. Rationalization is basic for leveraging IT functionality. It comprises the deployment of standard IT applications, new use of information content and making required changes to business processes, to maximize the benefits from the system functionality.

There are two types of integration involved on the rationalization level. The first is technical interconnectivity, which covers interconnectivity and interoperability of different systems and applications through a common IT platform. The second type is business process interdependence, which includes interdependence of organizational roles and responsibilities across distinct functional lines. (Venkatraman, 1994) Neither type alone is sufficient, but organizations allocated more attention and effort to technical interconnectivity than to business process interdependence, according to Venkatraman (1994). He continues to point out that internal integration must be guided by the needs of the market; simply fine tuning existing outdated process through current technological capabilities does not create the required organizational capabilities. Or as
Hammer (1990) once put it “Instead of embedding outdated processes in silicon and software, we should obliterate them and start over”. Hammer (1990) argues that IT should be used to radically redesign the business processes in order to achieve dramatic improvements in their performance. The degree of radicalism is the major delimiter between the first evolutionary level and the two revolutionary.

The first of the two revolutionary levels – business process redesign – includes efforts like the ones earlier suggested by Hammer (1990) and Johansson et al. (1993), but also redesigns among multiple participants in a business network. This level reflects a strong belief that IT functionality is not fully realized if superimposed on current business processes. (Venkatraman, 1994). The initiative and driving force of BPR has to be business related rather than technological, and emerge from business visions (Davenport & Short, 1990). As also argued on the rationalization level, the scope and benefits of business redesign, within or outside the limit of the organization, are broader than efficient transaction processing and must be cross functional as well as both business and technological oriented. (Venkatraman, 1994). Broadbendt et al. (1999) found in their research that boundary crossing IT-infrastrucure services were particularly important to BPR.

A fundamental question when discussing strategy is “What business are we in, and why?”. The third level, business scope redefinition directly addresses that question and what role IT has to play, if any. This level aims at redesigning the processes to such an extent, that the mere essence of what the organization does, its value proposition to its customers, is redefined. However, the term redefinition does not only refer to a single organization but aims at redefining the whole marketplace, changing the rules of the game (Venkatraman, 1994), as earlier laid out in the discussion about the productivity frontier and the use of BreakPoints.

Just as there is no single superior view on IT-infrastructure for all organizations, there is no one best level of IT enabled business transformation. Each level indicates a range of potential business benefits, greater benefits can generally be achieved at higher levels. What level can be reached is contingent on the organization’s ability to transform itself. The organization must decide on the rationale behind the initiative to redesign or redefine: “Is it to rectify current deficiencies or to create capabilities for tomorrow?” (Venkatraman, 1994) Venkatraman (1994) proposes two different roads or approaches to business process redesign, shown in figure Figure 13, as the two arrows. The left arrow symbolizes the avenue of seek efficacy, which focuses dominantly on automation of existing processes and resolving current weaknesses, while the current strategy is fixed and given. The right arrow symbolizes the avenue of enhance capabilities, which naturally aims to create strategic capabilities for future competition.

One of the most important decisions is whether to be at the evolutionary level or to be at one of the revolutionary levels, the decision is a choice of roads. Most authors (see for example: Porter, 1996; Davenport & Short, 1990; Broadbendt et al., 1999) agree that rationalization is insufficient in order to create competitive success and must sooner or later lead to some form of revolutionary act. However, it does not imply that automation is obsolete. The two roads are appropriate for different objectives and different times, and not necessarily mutually exclusive. As discussed above, Moore (2005) suggests streamlining context processes, that is, take the road of seek efficiency and at the same time heavily fund core processes in order to build the next generation capabilities, the road of enhance capabilities.
The view on IT-infrastructure is a critical issue in deciding on the desired level and what road to take as the IT-infrastructure can be one of the barriers and place massive constraints on process innovation. Process simplification, the road of seeking efficiency requires less infrastructure capabilities because the process changes are limited in scope (Broadbendt et al., 1999), and hence suits the none or utility view. Conversely, process innovation, the road of enhancing capabilities, requires more infrastructure capabilities because the process changes are more pervasive across the organization (Broadbendt et al., 1999), and hence oblige a dependent or enabling view. In conclusion, organizations with an extensive set of infrastructure capabilities experience fewer technological barriers to BPR implementation (Broadbendt et al., 1999).

8.2.5 Approaches to IT-architecture and complexity

Improving processes usually involves implementing systems and processes management across business functions rather than within functions (Broadbent, Weill & Clair, 1999). Business integration requires streamlined business processes and integrated information systems capable of combining information from many sources.

This hardly seems like an astonishing fact, but the history shows otherwise. In post WWII era the philosophy of decentralization was born to cope with increased complexity of continuous growing and diversified companies. The idea was to break companies apart in different autonomous (often product based) units, with their own management and little or no integration between. This worked fine until companies realized that serving customers efficiently required an approach that coordinates their internal efforts across product divisions and functions. (Markus, 2000)

Thus, most of the IT-infrastructure has evolved from addressing issues in a piecemeal fashion within separate business units. This has led to the current situation, where many organizations have a number of different system silos in place for different departments, with applications supported by discrete infrastructure platforms, which makes business integration and IT interoperability far from trivial.

A possible and relatively usual approach the last decade, especially in the product industry, has been to replace all old IT-systems with one centralized standard system and in doing so moving the responsibility of integration between the parts on to the supplier of the new system. This is often referred to as a rip-and-replace tactic, a route only available to organizations that have adopted a none or utility view on IT-infrastructure, since, as Davenport (1998) points out, these systems impose its own logic on a company’s strategy, organization and culture. The system represents a generic solution, which incorporates a series of assumptions on how business operates in general and what is best practice on a number of fields. In many cases, the system will therefore enable a company to operate more efficiently than it did before. But in some cases, the system’s assumptions will run counter to a company’s best interests (Davenport, 1998).

As a result, most companies installing enterprise systems will need to adapt or even completely rework their processes to fit the requirements of the system. For organizations sharing a dependent or enabling view it is simply not an option to hand off the control of the processes.

Another dilemma of the standard enterprise system is that it is developed to meet common standard requirements. Most of the time there is some sort of functionality gap between the standard solution and the user requirements (SAP, 2005). It has been estimated that in the best case, integrated enterprise systems only address about 70% of
the needs of the average organization. Therefore, the typical organization will need to buy additional new systems or retain older “legacy” systems to handle certain critical needs. Because business needs require these additional systems to work with the core integrated systems, everything will need to be integrated. Similarly, internal data cannot satisfy all of a company’s needs for analysis, so external data needs to be incorporated as well. (Markus, 2000) Hence, the requirement of integration remains with a centralized monolithic system.

Another aspect of the rip-and-replace tactic is that the so called legacy systems are generally wired into the running of a business in a very substantial way (Robertson 1997; Davenport 1998). This makes it almost impossible to substitute or replace them, at least over some considerable time. The issue has thus shifted from legacy migration, i.e. to move functionality from old to new systems, onto legacy integration, i.e. to integrate new systems into the existing systems, and further onto legacy wrapping, i.e. to keep and encapsulate old systems in order to utilize them in a new architecture (Coyle, 2000). Legacy wrapping goes hand in hand with another approach to decrease the fragmentation of the IT-infrastructure, namely a layered architecture that enables applications and systems to directly interchange information in a loosely coupled way.

The benefit of the heterogeneous architecture is first of all the ability to keep and exploit the legacy systems and the investments done in the past. It also presents the possibility to tailor parts of the IT-infrastructure and business applications especially important for the competitive advantage of the organization, as other non value adding parts can be standardized and bought of the shelf.

One could think of an architecture where standard third party products are used for context processes which does not differentiate the company and in-house developed functionality to support core processes. There is however obvious drawbacks with this kind of architecture, foremost the complexity that arises in the interface between components, which will demand a legible strategy for governance.

### 8.2.6 The source of complexity

The major difficulty of creating a comprehensive and well integrated IT-support is the unpredictable and changing nature of business. A well built IT-infrastructure works as a foundation for accommodating business applications and reduces the complexity generated by changing business demands on IT. (Broadbent & Kitzis, 2005) On a very fundamental level IT related complexity is a function of the amount of information the organization needs to handle, the number of sources it originates from and the number of relations between different pieces of information (Haeckel & Nolan, 1993).

If complexity in IT is unmanaged, it can become both expensive and obstructive to the business (Anderson et al., 2006). Yet, it is important to remember that complexity is a direct result of adjustment and change. The complexity of IT is generally not a result of errors, but a natural characteristic of a growing environment (Natis, 2005). Below follows a more detailed discussion on how complexity arises and what the source may be.

#### 8.2.6.1 Disjointed legacy systems

Independently designed systems have operated independently, unaware of other systems. Each has managed its own data, which accounts for only a fragment of the enterprise's overall information assets. The need to share information between systems drove the creation of integration links, but mostly in an isolated, opportunistic manner:
one link at a time. This left IT organizations with the overwhelmingly complex task of managing the resulting enterprise "spaghetti network" of point-to-point links between hard coupled applications: redundant, conflicting and unaware of each other. (Natis, 2005)

To be able to utilize the application landscape it is crucial to recognize that legacy systems are not relics of the past, but essential working components of the present. All systems must be treated for what they are: equally valuable contributors to the enterprise's information assets.

8.2.6.2 Business pressure

Many companies in different industries find themselves dealing with an over-abundance of custom designed products, services and IT functions. Companies quickly become trapped in a tangle of increasing production and service costs, reduced margins, slower time to market, and a clumsy multi-channel delivery process, in trying to meet the unique needs of every customer in every segment. The customer experience and operating efficiency is severely damaged. (Anderson et al., 2006)

IT organizations are under continuing pressure to achieve more with less – and to do so more quickly. This "more, faster, cheaper" syndrome naturally leads to poorly planned, rushed and under designed systems (Natis, 2005). Broadbent and Kitzis (2005) points out that as business executives have become more appreciative of how IT and systems impact their bottom line profitability and relationships with customers, they have become understandably more demanding. The amount of hype about what is possible with technology doesn't help. To reduce complexity caused by opportunistic, rushed or over engineered projects, companies need to foster a culture of respect for software architecture, of systematic software engineering and of realistic IT management (Natis, 2005).

8.2.6.3 Disruptive innovation

Continuing innovation keeps competitive enterprises ahead but also contributes to complexity. Most introductions of new technology follow the Technology-Adoption lifecycle, and depending on the IT-strategy, development will relate to the life cycle in different ways. Either as early adopters who, in search of innovative differentiation, jumps on the technology in an early market phase or more of a conservative approach, which waits until the technology has matured and commoditized. To pursue both goals, most large enterprises force some systems to stay on the safe, well understood side of their technology environment, while other systems are built using the latest, higher-risk technology offerings. (Natis, 2005)

There is no favorite of the two approaches, but they represent the extremes, two incompatible goals. It is essential to avoid innovation in isolation and therefore fundamental to deploy a big-picture-view on enterprise architecture to avoid an unmanageable complexity.

8.2.6.4 Neglected knowledge management

One source of complexity that should not be underestimated is a lacking ability to put the new technology and working methods to use. At the end of the day, employees are responsible for building and running the IT-systems. If the employees have not been given a chance to learn how to master new technology that is being introduced, chances
are good that they will utilize the new technology in the same manner as old the technology were used, and thus fail to realize any benefit from the technology change. Many companies fail to allow enough time for training, resulting in a workforce with outdated skills. (Natis, 2005)

8.2.6.5 Politics
Internal company politics often lead users to technology decisions that are motivated by considerations unrelated to technical quality or architectural merits (Natis, 2005). Vendor self-interest and politics, for their part, result in technologies designed to help the vendor as much, if not more than, they help the users. Much of the overspending, according to Carr (2003), is driven by vendor’s strategies. Big hardware and software suppliers have become very good at parceling out new computers, applications, and networking equipment much more frequently than they need to.

8.2.6.6 Mergers and acquisitions
Corporate consolidations create enormous pressure on IT departments, often forcing them to abandon their long-term architecture initiatives. For example, following a series of mergers in the early 1990s Elf Atochem found itself hampered by fragmentation of critical information systems among its 12 business units. Each unit was tracking and reporting its financial data independently and ordering systems were not integrated with production systems. The many incompatible systems made it impossible for information to flow smoothly through the organization, and top management was not getting the information it needed to make sound and timely business decisions. (Davenport, 1998, p. 128-130)

Assessing information systems is now a key part of due diligence in mergers and acquisitions and the success or failure of large IT-enabled business projects can now make or break a business (Broadbent & Kitzis, 2005). It is however important to notice that this is not an isolated technical issue. At Elf Atochem the executives realized the real source of the difficulties was not the fragmentation of its systems but the fragmentation of its organizations (Davenport, 1998, p. 129).

8.2.7 Managing complexity
Realizing that they are overwhelmed by more complexity than they can deal with, banks and other financial institutions are scrambling to streamline their offerings and operations. But simply eliminating complexity is a mistake. (Anderson et al., 2006) As mentioned earlier and illustrated above, complexity does not need to be the result of surplus activities and avoidable errors from the past. It could also arise from generally positive events such as innovations and mergers and acquisitions, so conquering complexity does not mean eliminating it, but rather succeeding despite its existence.

A high degree of complexity might on the contrary generate a sustaining competitive advantage, since the complexity makes the IT functionality hard to imitate for competitors. This of course rests on the presumption that the functionality delivers some kind of business value. Remember that competitive advantage or a capability does not rise from the IT-system per se; the question is how to use it to realize the strategic benefits. Thus, there is necessary and unnecessary complexity. Necessary complexity adds value for the company and the customer. It is the kind required to customize products and services and help companies increase revenues, profits, and customer
loyalty. Such customization does not constitute unnecessary complexity, at least, not by itself. (Anderson et al., 2006)

Complexity becomes unnecessary and value draining when companies fail to address the trade-off between customization and complexity, and the value derived from it. A trade-off between the costs associated with customization, and the price that should be charged for it. Complexity per se is not the problem. The problem is the inability to manage and control it. (Anderson et al., 2006) A first step in battling unnecessary complexity is to locate it. Then, progress depends on the organization’s ability to understand the problem’s origin and the organization’s IT-architecture. To maintain control of the information assets in an increasingly complex IT environment, organizations need to encourage enterprise-wide thinking across projects. Insist on understanding that each project is a contribution to the enterprise ecosystem, not an isolated undertaking. (Natis, 2005)

Sounds straightforward. But the challenge is to build an organization that routinely measures complexity and takes a continuous improvement approach to reducing it. This ensures that complexity is managed and customization that does not contribute to competitive advantage is eliminated.

*Toyota has become famous for their continuous improvement focus – a focus that can certainly transcend into other industries such as financial services. Creating a culture that consistently audits complexity and monitors the customization trade-off can improve both top and bottom-line numbers.* (Anderson et al., 2006)

If complexity in information technology is allowed to grow in an uncontrolled manner, companies may find themselves in a situation where IT becomes an inhibitor to new business. A strategic level awareness of complexity is therefore vital.

### 8.3 Business and IT perspective on tactical level

Statements articulated on the strategic level primarily aim to define long range goals and objectives in terms of market positioning and value proposition. Unless IT is an explicit part of the company’s value proposition, it is often left out. IT-managers, as well as other functional mangers, therefore have a need to concretize the strategy. This chapter will briefly discuss two different approaches managers commonly use to concretize business strategies to tactic level investment decisions.

#### 8.3.1 Management by maxims

Weil and Broadbent (1998) propose a management model where the strategy is specified in a number of business maxims and then further into IT-maxims. Business maxims express the common managerial understanding of which activities the corporation should engage in and on which level activities should be coordinated to capture potential business unit synergies.

Business maxims usually cover some or all of the categories *cost, value differentiation, flexibility, growth, human resources and management orientation*. Two sample business maxims from the second and third category are: “Develop customer partnerships on a worldwide basis.” and “Grow in cross selling capabilities.”. (Weil & Broadbent, 1998, p. 142)
IT maxims are derived from the business maxims. They serve the same basic purpose for IT managers as the business maxims do for business manager; they are guidance and a path to follow. Deriving the IT maxims means figuring out which implications the business maxims have for the corporation’s need to connect, structure and share information and how information technology should be deployed to assist in fulfilling business maxims. Relevant categories often used for IT maxims are: expectations for IT investments, data access and use, hardware and software resources, communication capabilities and services and architecture and standards approach. The IT maxims are often more concrete than corresponding business maxims and more often indicates implementation details. (Weil & Broadbent, 1998) Below are three sample IT maxims from category two, three and five:

- We need to have a common view of the customer across our business.
- We will have a common order entry system across business units that can cross-sell.
- We require data standardization for financial and sales data only.

(Weill & Broadbent, 1998, p. 87)

Using the IT maxims operational level efforts can be evaluated and justified for compliance with the company’s overall strategy. Projects and practices that do not fit into the maxim framework can be concluded not to be in the firms best interest and projects that are too massive to handle by a single business unit can be jointly initiated on a corporate level. (Weil & Broadbent, 1998)

8.3.2 Management by deals

Corporations that do not use management by maxims commonly use a technique referred to as management by deals. Deals based management is a more short sighted approach of deriving operational level IT investments from company strategies. Essentially corporations using deals based management refers all their IT investment decisions to business units. Since business unit managers are usually more short-sighted and result oriented than corporate managers, this often results in a more restricted and fragmented IT capability. No single business unit is usually willing to take responsibility for a proactive build-up of corporate wide infrastructure, which restraints exploration of future business opportunities. Infrastructure investments are based on political agreements – deals – between business units in a reactive manner and are justified by business cases rather than strategy. Managing by deals implies taking a none or utility view of IT. (Weil & Broadbent, 1998)

8.3.3 Anchoring IT investments in strategy provides an advantage

Corporations wanting to take a dependent or enabling view of IT must make sure there is long sightedness in IT investments. Complexity, as previously noted, often arises from short sightedness and limited consideration of future needs (Natis, 2005). Managing by deals is an inherently short-sighted approach to IT investment that threatens to feed complexity in the IT-infrastructure. Therefore management by maxims is preferred for corporations trying to use IT as a strategic resource.
8.3.4 Managing complexity

In the earlier discussion about complexity it was concluded that complexity is becoming such a massive problem that organizations should consider taking a strategic approach to managing it. In doing so it is vital that IT maxims are formulated in support of reduced complexity. Corporations willing to let complexity grow unmanaged will sooner or later find their strategic flexibility diminished.

8.4 IT perspective on operational level

The bottom level of the strategy pyramid consists of the operational level. It is at this level plans are carried out and problems solved. To reduce the complexity which arises from disparate IT-systems an architectural approach is needed, but not sufficient. The following chapters will cover how a Service-Oriented Architecture can be used to reduce complexity.

9 An IT-architecture designed for change

Architecture is commonly referred to as the art and technique of designing and building, as distinguished from the skills associated with construction (Encyclopædia Britannica, 2007). When realizing IT maxims, formulated as tactical objectives, on an operational level the skills of software construction will naturally be crucial. Programming languages needs to be fully mastered, quality assurance processes must be in place and so on. However, before the actual construction starts one need to design the software, or the whole application landscape, and make sure that it will work as intended. Since much of the complexity, as shown in the previous chapter, is caused by applications never designed to interact with other applications, a set of good software architecture principles have real potential to put words into action when it comes to reducing complexity.

This chapter will discuss one type of software architecture, Service-Oriented Architecture, and its’ potential for reducing complexity and increasing the value of the organizations IT investments.

9.1 Isolating logic in components

In a previous chapter, we discussed the approach large enterprises have taken to manage their organizational complexity. The idea was to divide the enterprise into autonomous business units, with only a loose interdependence accomplished through the corporate head office. The software community has for years worked with similar approaches to break down large monolithic software into manageable units.

Object-oriented programming is one well-known technique, aimed primarily at reducing complexity of development by simplifying code reuse. However, objects are mainly a concern for programmers developing new software, thus not addressing the concerns of those who are unable to build software from scratch. The desire to acquire manageable chunks of prefabricated software is addressed by Component Software. The idea is to compose the desired functionality out of a set of prefabricated components. (Szyperski, 2002)

The key characteristic properties of a component is, according to Szyperski (2002), that it constitutes a unit of independent deployment, can be used in compositions by a third
party and has no externally observable state. These three requirements imply a number of properties that components will exhibit.

First of all, the requirement that a component should be a unit of independent deployment implies that the component itself is a piece of ready made software that only needs to be configured and started to be operative (Szyperski, 2003). In turn this means that the component will have the properties of being well separated from its environment, i.e. encapsulated, and that it will be deployed in full or not at all. The second requirement gives that the component can be used of someone who does not know how it was built and the third requirement implies that a component cannot be distinguished from copies of itself. (Szyperski, 2002)

The rationale for components is that the concept introduces a viable option for acquiring IT capabilities, without neither buying a packaged application nor building the entire functionality from scratch. Szyperski (2002) argues that packaged applications often turn out to be a too inflexible and imposing to represent a good alternative. The author concur with Davenport’s (1998) analysis, discussed earlier, that enterprise systems, which is a form of packaged applications, by their inflexibility threaten to erode competitive advantage for adopters and therefore should be avoided. Almost equally bad, however, is to attempt to build the entire functionality oneself, since this will be infinitely expensive. By composing desired functionality out of unrelated components, one can leverage work completed by a third party, without committing to any particular structure or process imposed by this party. Ideally, compositions consist of both bought and custom-made components, in order to get the best of both worlds.

The key benefits from componentization are derived from reuse. In accordance to the above argument time and cost for development will be reduced, whilst flexibility will be maintained, if components are used. The cost for reusing a component has been empirically shown to be 10-60% of implementing a new one with the same functionality (Lim, 1994). The majority of findings are in the range 10-20%. Cost savings have largely been explained by an increase in productivity, since the speed of the development cycle increase accordingly (Lim, 1994). Other favorable aspects of reuse are that the more users a component has, the more quickly the numbers of defects are reduced (Pfleeger, 2001).

Combining flexibility, high pace development and low cost of usage; software components have a great opportunity to tackle several of the sources of complexity. Compositions enable developers to quickly assemble new systems in a more structured way and move attention from low-level implementation details to design and architecture, thus helping to manage complexity caused by business pressure. Over time, components can be updated individually as more mature versions become available, reducing the risk of introducing discontinuous technology in some components and the complexity related to disruptive innovation.

Unfortunately, nothing comes for free. The relative cost to produce reusable components have been estimated to 111-480% of non-reusable software (Lim, 1994). Another factor to take into account is that the robustness of the composites will inevitable decrease with more components used in a single composition (Szyperski, 2002). In order to gain the advantages associated with reuse, developers must “outsource” many functions to other components or the underlying system environment. The contextual dependencies on the underlying environment add to complexity and must be managed carefully. One way of managing this problem is to work to reduce the number of static dependencies, i.e.
dependencies that cannot be reconfigured (Szyperski, 2003). Some dependencies are more or less unavoidable, for instance a component written in Java will always require a Java virtual machine, whilst specific interdependence between components should be managed carefully.

The degree of coupling is a measurement on the interdependence between components. The effect of having “highly” or “tightly” coupled components is generally that a change in one component will affect the workings of other components, i.e. the other component will stop functioning or deliver a faulty result. (Pfleeger, 2001, p. 220) In terms of software development and maintenance hard coupling result in complex systems, as it is usually hard to find and understand all the links between different modules once the software has been written. Incorporation of even a minor change in a hard-coupled component will generally result in extensive modifications of related components. (Pfleeger, 2001, p. 220-222; Natis, 2005) Coupling cannot, however, be completely avoided if components should be able to communicate. The optimum sought for is therefore not uncoupled, but loosely coupled, components (Pfleeger, 2001, p. 220-222; Erl, 2005, p. 297; Newcomer & Lomow, 2005, p. 75; Dromey, 1996).

Loosely coupled components are dependent on that they communicate some piece of information, but are in other aspects independent. There are three types of coupling to consider: interface, technology and process coupling. Interface coupling refers to the requirements an information, or service, providing component sets up for callers to adhere to; does a caller, for instance, need to name arguments in a given order or have knowledge of how the provider internally have organized data? Technology coupling represents how technology dependent the component is. Some components depend so heavily on functions provided by, for instance, the type of computer they are running on, that they cannot communicate with a component running on a different kind of computer. Process coupling, at last, is the dependence on being a part of a specific (business) process. A component may be specified in a way so that it only makes sense in the context of a given process. (Newcomer & Lomow, 2005, p. 76-77)

9.2 Service-Oriented Architecture

Components represent reusable units of logic and immutable data. Organizations, however, often have a need to build structures for reuse of both logic and dynamically created data, that is, they want a running instance of a component. For example, a component capable of storing customer data will be of marginal use, unless it actually contains some customer data. Ideally, enterprises would like to have its customer data stored in one single place that can be accessed and manipulated from any device in any business process. This entails having, not only a unit of deployment, but rather a running service. The easiest way of envisioning a service is to look at it as an instantiated component; a component that is already running and maintained by someone (Szyperski, 2002, p. 470).

Services are also units of composition and the act of composing them are commonly referred to as building a Service-Oriented Architecture – SOA. Below are the fundamental principles of SOA laid out. The descriptions are based on what have been perceived to be the most common understanding of the acronym and there is no doubt that other definitions exist. It is important to understand that the main drivers behind SOA are much more business oriented than those behind component software. Therefore, the emphasis will not be placed on technical issues, but rather on the foundational
concepts and how they relate to what has previously been discussed in this thesis. The reader should also notice that the term SOA is used both to refer to the architectural ideas as such and to a system realized using these ideas.

9.2.1 Encapsulate logic for an individual step in a business process

One fundamental difference from other, earlier, software architecture paradigms is that SOA is designed to reflect real world business processes and activities (Erl, 2005; Newcomer & Lomow, 2006). Recall the definition of a process as “a set of linked activities” (Johansson et al. 1993, p. 57) and how in each activity some processing is done to further add value to the finished output of the process. The general idea of SOA is to structure IT in the same fashion as business processes, with a set of services that are linked together to form a process. Each service includes a distinct piece of processing logic, which takes input and produces an output, which in turn can be further processed by another service. Ideally services are modeled to directly support individual business activities, for instance a bank might have a “balance check” service that takes some piece of customer information as input and returns an account statement as output (illustrated in Figure 14). Furthermore, an ideal SOA-service can, without modification, support several types of users involved in different types of business transactions, but with the same basic computing or information needs. The service is simply reused and support several processes at once, as seen in Figure 15. (Erl, 2005; Newcomer & Lomow 2005; Lopez & Peppiatt, 2005)

![Figure 14 Example of Check balance Process](image1)

![Figure 15 Reused service in another process](image2)

Except from aiding reuse, the rationale behind capturing individual steps of a business process in services (instead of automating the entire process in one big application) is flexibility. If the process changes, the SOA will change with it. If a process step is reduced, for example in a business reengineering effort, the service which support that step can, in a slightly simplified world, just be shut down or replaced. The process can be
rebuilt without extensive programming efforts only if a number of smaller services, each with well-defined scope, support the process. (Erl, 2005; Newcomer & Lomow 2005)

Envision for example the shipping process presented in Figure 16 containing a number of steps including wrapping of the goods, preparing customs declaration forms, preparing a waybill and loading the shipment on to a transport. Due to deregulation in the European market the customs declaration step becomes obsolete and the process is shortened. If, however, the IT support system still insist that personnel accurately enter the information needed to complete the customs declaration, before proceeding to shipping step, very little will be gained from the deregulation. Companies engaging in business process reengineering efforts usually have far more than the exemplified single step to remove or simplify in their processes. It is easy to see how a Service-Oriented Architecture in these situations has a potential advantage to older and more rigid ways of building IT support systems.

![Figure 16 Example of Shipping Process](image)

### 9.2.2 Two Service types, entity or process centric

Business services can essentially be built in two ways; process centric or entity centric. Process centric services are services that embody some sort of process flow logic to support a specific business process and have operations that relate to execution steps in the process. Entity centric service, on the other hand, are services that perform operations on some given entity of data, such as a customer record, an arrangement or an account. (Erl, 2005, p. 399-480)

A process centric *money transfer service* may for instance have a method, in support of an Internet banking application, named *Move Money Between Accounts* that accepts two account numbers and an amount of money to be transferred in between them. The entity centric equivalent would be an *account service* with the operations *Put In* and *Withdraw*. The entity centric service would have to rely on either an application service or a process centric service to provide the process logic (i.e. to call the two operations in the correct order), but would on the other hand be much more flexible and reusable than a process centric alternative. The bank might for instance in addition need a transfer
service, for use by tellers in the office, that debits a service fee from the customer account. If one originally opted for the process centric solution a separate service will have to be written to handle the office application; but if one, on the other hand, chose to implement the entity centric solution only the processing logic will have to be rewritten.

In order not to make the business services too difficult to develop and coordinate, a framework of supporting services needs to be in place. Functions like service catalogues, orchestration coordinators and policy controllers do not in themselves contain any business logic but are essential to the realization of business services and should, in accordance with SOA principles, be implemented as services. Another class of services that belongs in this category is services that constitute interfaces or “bridges” to legacy systems. Together these two classes of services are often referred to as application services. (Erl, 2005, p. 333-341)

9.2.3 Deriving service candidates

Deriving business services can, according to Erl (2005), be done in one of three ways. Top-down, bottom-up, or a combination. The bottom up strategy takes its starting point in existing architecture and with existing applications. Services are built as needed and existing systems are encapsulated in services. The main driver behind a bottom-up application of SOA is usually integration aspects, i.e. companies are building services to make exchange of data between different systems easier. The problem is that this approach often fails to create fully SOA supported processes, since it mostly focuses on the exchange between applications. (Erl, 2005)

The top-down strategy takes it starting point in a complete enterprise business model, that is a complete description of all processes in the entire enterprise. From the model a Service-Oriented Architecture is composed by identifying process steps suitable for encapsulation and automation. Since the analysis is based on the complete set of business automation needs, it is with this approach possible to fully leverage benefits from service reuse and create a highly flexible architecture. Unfortunately, the upfront analysis work needed to take this approach is usually very extensive. To accurately capture every activity performed on every level in every business unit is a major undertaking in most organizations. In consequence, this approach is only feasible for very small organizations. (Erl, 2005)

The last approach is the meet-in-the-middle approach. Instead of performing a complete enterprise wide process analysis before building the first service, work starts in priority areas. The priority areas, or processes, are analyzed using a top-down approach and services are implemented as soon as the analyze have covered the area. Analysts then move on to other areas with lower priority and the process is repeated. The obvious benefit of this approach is that it is more manageable than the pure top-down approach, but still produces service support for full business processes. The detriment of choosing this approach is that services may need to be revised, or implemented twice with slightly different functionality, as the work progresses, since it is hard to fully generalize to capture similarities with processes not yet analyzed when doing the initial set of services. (Erl, 2005)

9.2.4 Operations and messages

Each service in an SOA is a collection of operations. An operation is the callable part of a service. A customer-data service may, for instance, have operations like
StoreCustomer, FindCustomer and EraseCustomer. The inputs and outputs of operations are called messages.

A message can be defined as an autonomous unit of information. It contains one data part, carrying whatever information the sender wishes to convey, and one metadata part with information needed to route the message to its correct destination together with any information the receiver will need to successfully process the message. SOA recommend that messages should be as self sufficient as possible; holding a logic entity of information in its data part. For example, sending one order as a message is preferred to sending individual order lines. (Erl, 2005; Newcomer & Lomow 2005)

Electronic messages can be composed in many ways. E-mail is one form of electronic messages and displays their content as plain text in human readable language. Computers, however, generally have difficult to understand human language and often use a binary format, not understandable by humans. SOA stipulates use of the XML-format for messages, since it can be read and interpreted by both humans and computers (Erl, 2005; Newcomer & Lomow 2005). XML-data is described by tags, indicating the type of the data, and ordered into tree structures to indicate relation between different types of data. Below is an example order:

```xml
<order>
    <customer>M.Jordan</customer>
    <orderline><item>Basket</item><quantity>2</quantity></orderline>
    <orderline><item>Ball</item><quantity>1</quantity></orderline>
</order>
```

Reading the message it is easy to deduct that this is one order, placed by M. Jordan, for two baskets and one ball. The computer does not understand English but can separate one tag from another and be instructed on how to handle the content. A special schema can be devised to describe which tags that are allowed in a specific document and how they can be combined.

SOA prescribes that all messages should be standardized to form a common data model. As implementations grow larger and comprises more and more systems it is vital that all involved parties have a clear and common understanding of message content. (Newcomer & Lowmore, 2005, p. 73) If, for example, the element named “price” in one system indicates the price excluding VAT and shipping, whereas it includes those items in another system major confusion is bound to arise. One cannot a priori establish that one way of entering prices is better than another, or assume that everyone has chosen to take the same approach; it must be explicitly agreed on and documented before systems even try to exchange data. (Newcomer & Lomow, 2005, p. 119-121) This is not, however, to say that systems should have a common internal data representation; systems can use whichever data model they wish as long as they use the agreed definitions when communicating with other services (Erl, 2005).

### 9.2.5 Service contracts as detached interfaces

Formal contracts defining services in a both machine and human readable format is one of the cornerstones of SOA. Both Erl (2005) and Newcomer & Lomow (2005) argue for messages based on XML and the use of a standards based metadata framework for specifying contracts. The idea is that each service should only expose a small and well-defined service interface. A caller should not have to know anything about the internal
workings of the service in order to call it, as long as it can provide the input data and comply with the policy rules stated in the service contract. (Erl, 2005, p. 33-50) Key benefits from using this approach is:

- The formal contract simplifies usage since it clearly specifies what operations the service provides, the format of the input and output parameters, as well as any policy constraints that applies.

- Basic validation and conformance checks can be automated and built using standard XML-validation tools. This removes the need for generic validation logic at both server and client side, which substantially simplifies the task of for example adding extra functionality.

- On the client side many developer tools now support generation of stubs from XML service definitions. This substantially reduces development time and simplifies the task for developers using a given service for the first time.

- The machine-readable contract can also be interpreted by humans, reducing the need for separate documentation. Although users are warned that XML service definitions can become quite complex and does not necessarily convey the semantic meaning the described entities.

Besides defining delivered functionality, inputs, outputs and constraints of a service, the contract also defies how to contact a service. The de facto standard language for service contracts is WSDL. One strength of this language is a strict separation between the abstract definition and implementation details, such as the address of the service, is enforced. It is therefore possible to add such information as needed, if for example a given service is accessible using both SOAP over HTTP and SOAP over IBM MQ. A service contract can also contain data about for example availability. (Newcomer & Lowmore, 2005, p. 109-117)

Service contracts should be stored in a central repository. The rationale behind this is that services easily can be discovered and evaluated by developers prior to usage. Since descriptions are machine-readable, they can be searched and matched for certain criteria. (Newcomer & Lowmore, 2005, p. 73; Erl, 2005, p. 248-256)

One obvious benefit of creating a detached interface of the service is that it can be more easily replaced. One might, for instance, decide to replace a homebuilt component with a third party component that offers the same functionality to a lower cost. Since the service consumer identifies the location of the service producer by looking at the contract, exchanging the producer can be fully transparent to the consumer. (Erl, 2005) The concept is very similar to the object concept in object-oriented programming. The main difference is that objects encapsulate both logic, data and state, whereas services encapsulate only logic and store data and state in the messages sent back and forth between services.
9.2.6 Services are stateless

State information is defined as information related to current processing that can be tied to a particular service requestor\(^2\). Services should as a general rule be designed to minimize the use of state information, and in particular not use any state information prevalent between subsequent invocations. (Erl, 2005, p. 307-308; Newcomer & Lomow, 2005, p. 82)

The main argument for stateless services is scalability. One part of the argument is that heavily used services simply can be overburdened if they have to keep track of every single client session from every single client that is accessing them. In such cases one generally wants to scale horizontally, building several identical services, but this is also inhibited by statefull services since only one of the instances will know about each client’s current state. If executions are lengthy with several invocations clients cannot be redistributed between instances as needed but will instead be tied to the instance they initially contacted. Other related issues may also arise; a service instance may for example be required to verify with all other instances that no processing for a particular client is currently under way before starting its own execution, which of course will increase to overhead. (Newcomer & Lomow, 2005, p. 82)

The approach taken by SOA is to append current state information in the header part of each message (Erl, 2005, p. 307) or, in special cases, rely on dedicated services to keep certain kinds of state information (see for example single-sign on description in: Erl, 2005, p. 260-261).

9.2.7 Services are invocation style neutrality

Reusability of both business and application services can be greatly improved if services are designed to support multiple invocation styles, i.e. if they for instance can handle all three of batch processing, request/response invocation and asynchronous queuing. The business function performed by the service will usually be equivalent regardless of how a requestor wishes to call it, but the implications to the requestor may be significant. (Newcomer & Lowmore, 2005, p. 81-82) A service that can only invoked by once-a-day batch processing will be of little use to someone developing an interactive web portal that requires a response within a maximum of 1 second.

9.2.8 Services are standards based

Services should except from being accessible using different invocation styles preferably also be accessible from all technology platforms. In order to achieve these common goals, support by multiple vendors is essential. There are several in-place, or emerging, standards that define different aspects of SOA functionality. Since the standards are open and backed by most major vendors, a choice of implementation is available. A very brief overview:

\(^2\) Example: For a webmail server (such as webmail.student.liu.se) information about which user is logged in from a particular computer and which folder the user is currently browsing constitute state information. If the server did not keep track of such information the user, for instance, would have to provide his/her password for every single mail he/she choose to open. The actual mails, on the other hand, would not be considered state information, since they are persistent and exist unconditional of any particular login or client computer.
• SOAP – The foundational communication protocol for web services. Provides the ability to call and receive web services.

• WSDL – Metadata language standard used to describe a service.

• WS Addressing – Standard for addressing calls to web services. Specifies source, destination, reply address, message id, relationship with other messages and so on.

• WS ReliableMessaging – Guarantees that a sender will be notified whether a message transmission succeeded or not and that fragmented messages will be delivered in order or generate an exception.

• WS Policy – Makes it possible to attach properties, for instance rules, requirements or preferences, to a web service in the form of assertions that are applied to each message.

• WS Metadata exchange – A standard for programmatically retrieve metadata about services.

• WS Security – Framework for adding identification, authentication and authorization to services.

• WS Notification and WS Eventing – Two competing frameworks for implementing the publish-and-subscribe messaging model.

• WS BPEL – Language for encoding business processes as data flows between services.

• WS CDL – Specification for information exchange between multiple organizations.

The above list is neither extensive nor exhaustive; the point is that there is already a standard for almost every conceivable aspect of SOA, so the need for homegrown solutions should be small. It should, however, be noted that many of these standards are yet under development and only a few has been unanimously embraced by the IT community. (For an more in depth description of available standards see for example Erl, 2005)

9.2.9 Reuse and composition of services

Since services can be viewed as a subclass of components, they inherit the advantage of reuse. Reusing services should, generally, be less expensive and take less time than building new ones. It is, however, important to realize that in order for reuse to take place the service must in accordance with the above reasoning, be modeled to have a meaning in more than one business process. The process coupling can be reduced if the service is made less activity-specific, something that can be accomplished by adding processing instructions to the service’s input message (Erl, 2005, p. 292-293). For instance an invoice submission service may be made less activity-specific by adding an input field which states how the customer wants the invoice delivers (by mail, email, etc).

Just as software components, services can be composed. In contrast to components, however, services are composed at runtime and not at deployment. A composite service is
simply one service which is programmed to call other services to do some kind of computation or data retrieval. (Newcomer & Lowmore, 2005, p. 232-269)

Envision how the example account service, described earlier, invokes a legacy subsidiary ledger system wrapper, a transaction log service and an account history record service. The legacy account service in the example is not built on SOA technology but still be fully functional, very reliable and mission critical but when wrapped in an SOA application service it can still be a part of the composition, together with two other services that offer extended functionality that is also reused in other places.

The above is an example of how a composite service is used to implement a small business process, account update, consisting of the steps execute transaction, record account history and log the two events. Such composite services can be built programmatically, that is they can be written in a programming language such as Java, but they can also be implemented using an orchestration engine. Orchestration is an SOA term commonly used to reference a composite created by configuration, rather than programming. The difference may seem insignificant, but may have a great impact on the development process. Creation of the WS-BPEL definition, which makes up the configuration of the orchestration engine, can usually be aided by graphical design tools visualizing the business process that is being implemented. The effort needed to update a definition file should also be considerably less than that of rewriting a programmatically created service, adding to the flexibility of SOA. There are, however, a number of downsides. One is that WS-BPEL is not yet mature, nor is the implementation of the orchestration engines. Performance may also take a big hit from using what is essentially a scripting language to automate transaction intensive processes. (Newcomer & Lowmore, 2005, p. 232-269) The greatest problem is most likely, however, that orchestration presupposes services that can already communicate seamlessly.

9.3 The unexplored domain of Service Granularity

The above presentation of components and Service-Oriented Architecture is by no means complete and only designated to provide the reader with an overview of the fundamental concepts of SOA and architecture. The literature provides a rich overview of different technical aspects and also discusses different approaches on how to go ahead and implement SOA. There are however aspects that the SOA specific literature does not cover, or attempt to cover. One of these is the question of how abstract services should be; how large domain of functionality should a single service cover? The assumption seems to be that with the right intentions and a well-appointed process this question should work itself out. Newcomer and Lowmore for example states:

Principle #5 – Services should be defined at a level of abstraction that corresponds to real world business activities and recognizable business functions so as to better align business needs and technical capabilities.
(Newcomer & Lowmore, 2005, p. 73)

If the top-down approach to SOA implementation is taken, the richness of information to ground decisions about service granularity on, may very well be enough to avoid any problem related to this issue. Using any of the other approaches may on the other hand result in a situation where an arbitrary decision will have to be made.
10 Dimensions of service granularity

As argued above one well-recognized way to cope with complexity in an IT-architecture is to use a divide-and-conquer strategy. The complex architecture is divided into components which are realized and published, internally and/or externally, in the shape of reusable services. In a scenario where full information about all possible usages is known, i.e. the-bottom-up or meet-in-the-middle strategies, an obvious question becomes: what services should be realized? Which in turn, of course, depends on what should be done, and for whom?

10.1 The extent of a service

A common way in order to capture users’ preferences and expectations on a system is the use of use cases. A use case captures a contract between the stakeholders of a system about its behavior (Cockburn, 2001). One of the most important elements of a use case is that it describes what goal actors have, in other words, who wants what done. What has been said so far is in very general terms; do not be fooled into thinking that use cases are only applicable in the IT domain. It applies equally to business and computer systems. The actors can be individual people, organizations or computers, or a mixture of them all (Cockburn, 2001).

One of the foremost important but also hardest tasks when writing a use case is to decide on what to include and what to exclude in the use case, i.e. to set the boundary for the use case. The scope of a use case is the extent of the design it covers, once again IT or business. Cockburn introduces an absurdly simple, yet remarkably effective tool for defining the scope, which he calls the in/out list. It is simply a table of three columns labelled: the topic, in and out. Whenever there might be confusion whether a topic is within or outside the scope, add it to the list and ask everybody concerned if it is in or out. The amazing result is that while it is clear to each person whether the topic is in or out, the views are often opposing, which illustrates the need for explicit definitions. (Cockburn, 2001) Cockburn continues to define three main categories that always have to be explicit and unambiguous defined:

- The scope, what is really the system under discussion? What is the goal?
- Primary actor, who has the goal?
- Level, how high- or low-level is that goal?

It is however not enough just to satisfy the actors that have direct goals or demands on a system, or for that matter a service, the entire architecture of services must come into consideration. To what extent should services be offered? All three of the above categories must therefore be evaluated, for each service.

10.2 Three dimensions of service granularity

An architecture, or corporate IT-platform, can be defined in two dimensions according to Keen, in terms of reach and range (1991). Reach refers to the locations a platform is capable of linking, or in other words to whom is it possible to connect? Range refers to the degree to which information can be directly and automatically shared across systems and services or what services can be shared? (See Figure 17) The dimensions of reach and range determine the future business degrees of freedom i.e. which IT-dependent
products and services that are practical to implement and the architecture will therefore enable or disable future business options. (Keen, 1991)

Figure 17 Reach and Range (adapted from Keen 1991 and Weill & Broadbent, 1998)

Weill and Broadbent have applied the concept of reach and range to investigate what kind of IT-infrastructure services organizations have, and to what extent. They define reach much like Keen, as the locations and people the infrastructure is capable of connecting. Range refers to the functionality in terms of the business activities that can be completed and shared automatically and seamlessly across each level of reach. (Weill & Broadbent, 1998)

As argued above it is not just a question of determining and producing the by business requested functionality, but more important to manage the complexity that arises as the reach and range increases. One way to approach this is by dividing the architecture in components, or services. A key issue in doing so is the need for explicit definitions of a service, as illustrated by Cockburn (2001). In order to avoid ambiguity, the functionally notion must be partitioned and expressed in terms of how much functionality and what kind of functionality.

Neither Keen nor Weill and Broadbent concretize the concept of range. However, Weill and Broadbent provide an example of a service range also illustrated Figure 17: from just sending a message, on to access to stored information, performing autonomic transactions, and lastly to perform complex transactions on multiple applications. This example clearly demonstrates the increased functionality but tells nothing about what kind of functionality that is considered. What kind of functionality can therefore be seen as another new dimension to the reach and range model and will be defined in terms of realm. If realm is not defined, and thus left unanswered, chance will govern and it will be almost impossible to achieve any kind of intended reuse.
Consider the example of varying range but undefined realm presented in Figure 18: Low range of an inventory service could be sending an e-mail to check current inventory, a bit more increased range could be to publish inventory status on the web, on to a higher range of actually making the order online, to finally making the order online and getting exact delivery details such as shipping fees and dates. It is obvious that the functionality increases but nothing is stated about what kind of order is at question. It could be a customer order, although just as well a supplier’s order. With the purpose to define a service the definition of *realm* of the service, as those of *reach* and *range*, becomes a crucial task. It is not obvious that customers and suppliers should have different order services, nor is it apparent that they should not. One might suggest that a proper separation would be one service for standard orders of high volume, and another for low volume of complex orders.

The size or extent, the granularity of a service, will from here on be defined by all three dimensions simultaneously, that is, in terms of reach, range and realm and will be referred to as the R³-model. The granularity is visualized by the spatial extent, the volume that the orthogonal axis demarcates as seen in Figure 19.

The allegory of coarser grained services derives from the size of minerals such as rocks and stones. A gravel path is made up of coarse-grained gravel and each individual grain is larger than that of, for instance sand on the beach. The sand is more fine grained, which is illustrated in the case of services in Figure 20.
Another example of varying realm is a bank’s party service that stores and manages party information. What kind of parties should be included in the service? An obvious one may be customers, but what about the bank itself? Relevant data such as addresses and number of workplaces needs to be stored and managed as well. One approach is to provide a single service for both kinds of parties as seen to the right in Figure 21. Customers may however be of different types such as individuals and organisations, which in turn can be either public or private. Should all these different kinds of parties share the same services?

Figure 20 Granularity of services

Figure 21 Examples of realm

Figure 22 shows an example of two different look-up services of the same granularity, both accomplish the same functionality but for different service consumers and on different kinds of entities, hence different combinations of reach, range and realm. Notice that the volume is equal in both pictures, thus same granularity.
10.3 Limitations in spatial extent defines service granularity

It is apparent that business processes matter, and that organizations want to be able to be capable of quickly modifying a process when necessary. However it is important to remember that the problem for companies striving for business and IT alignment is far greater than just overcoming technical systems incompatibility. The challenge of creating a modular plug-and-play architecture that can be reconfigured at will to support business strategies goes far deeper than the IT organization, and reflects the business strategy perspective at the center of the organization (Luftmann, 1996).

Nevertheless, integration is inevitable. Every large firm in every industry has seen how competitive and operational necessity has made it essential to put more and more processes online (Keen, 1991). Many organizations have a number of different systems silos in place, for different departments, with applications supported by discrete infrastructures and platforms. This has historically not been a problem since the need for interchange has been small. But at times of just-in-time everything, cross-functional coordination and customer and supplier cooperation, it is almost certain that at some point these core business services will need to link together. IT can at that moment become either a business enabler or a disabler.

“Innovative business process can be delayed by the lack of information technology infrastructure services.” (Weill & Broadbent, 1998, p. 87)

If no consideration of time, money and effort had to be taken, the proactive organization would of course wish that the architecture as a whole had as much reach, range and realm as possible. However, time, money and effort are important considerations and the resources are of course limited. Reaching everyone with all the functionality of every kind, in order to keep as much future business degrees of freedom as possible is not always the path to business excellence. It depends on what kind of view the organization has on IT-infrastructure. As Weill and Broadbent points out (1998, p. 87): “No one view is best for all”, infrastructural capability is a strategic choice. No volume fits every organization at every time. The different views have different impacts on an organizations performance.
The extent of reach, range and realm, the volume, of the entire architecture is thus set by the organization’s view on IT-infrastructure. An important observation is that there exist more than just one set of functionality in the infrastructure. If for example a certain functionality realm of some kinds should access stored data to anyone, anywhere, it does not mean that all business functionality of the infrastructure has to reach anyone, anywhere. Some other single kind functionality could for instance do complex transactions within a single business unit. There is furthermore no necessary relationship between that volume of the architecture, and the volume of an individual service. If every single service has a small volume, i.e. low reach, range and realm respectively, there has to be more services in the architecture in order to perform the same business functionality. The volume has to be constant in the architecture as shown in Figure 24 since the demands by the business will not disappear due to implementation choices. The reader should however keep in mind the difference between implementing one single service to meet the demanded functionality and implementing several. There will be more inter-dependencies between services in the case of several services.

**Figure 23 R³ of the entire infrastructure**

**Figure 24 Same functionality, different service granularity**
There is furthermore no right volume for services in the architecture, each service has to be evaluated individually. The volume can be altered by changing the extent of each dimension respectively. It is however important to remember that the dimensions are not independent to one another, an increase in reach can have significant impact on range. For example, if the reach of a service is extended from domestic to serve the organization globally, the functionality offered may have to decrease in order to not let development and maintenance expenses run wild. There is, as illustrated by this simple example, often a trade off between the dimensions. Hence, it is extremely important to consider all three dimensions simultaneously when defining a service granularity. One tool in order to clearly define what is within the service, inside the volume, is the in/out list presented by Cockburn.

Moreover, it is hardly a simple task to realize a specific volume, or service granularity, in terms of the R³-model even when one has a clear understanding of the organization’s view on IT-infrastructure. Weill and Broadbent (1998) recognize four major inhibitors to achieving the desired reach and range:

- The incompatibility of different computers, operating systems, standards and propriety applications.
- Different data definitions, semantics such as customer information, and different technical standards in several applications in various parts of the organization, making systems incompatible.
- Politics and power issues across different areas within the organization.
- The reluctant of senior management to provide the substantial up-front dollars required to invest in an extensive infrastructure.

These inhibitors are also true for the R³-model since range and realm in terms of the R³-model is equivalent to range in the vocabulary of Weill and Broadbent. The reader should also notice the similarities with the earlier presented causes of complexity in an IT-architecture (see chapter 8.2.6). This is of course due to increasing the volume of a service, that is decreasing service granularity and making the service coarser grained, increases all dimensions of reach, range and realm, resulting in a more complex service.
11 Specification of the task

The relationship between the concepts described in the literature review is cyclical as seen in Figure 25. The strategy today will define the degree of business freedom the company will enjoy tomorrow, through the route of either maxims or deals. The investment decision on what infrastructure functionality is needed will govern what services will be realized. How the services are implemented in terms of R^3 will define the architecture complexity, which in turn will limit the degree of business freedom, thus limiting what strategy is feasible in the future.

**Figure 25 Method of analysis**

Customer demands, articulated and inferred, and a company’s strategic view on IT will set the rules of the game, as well as regulatory demands. The strategic view is internal i.e. it can be defined by the company itself in contrast to the customer demands. However, the strategic view will influence many steps in the decision making process and is not easily transformed and hence restricts possible strategies.

There are two distinct approaches to concretizing the business strategy into Infrastructure Functionality: the route by maxims or the route by deals. If the route by maxims is used, the business strategy is concretized into business maxims which are expressed in a more concrete manner as IT maxims – how information technology should be used to support the business strategy. The IT maxims, in turn, define specific functionality the IT-infrastructure must cover, in terms of R^3 – reach range and realm.
The route by deals converts the corporate business strategy into Business Unit strategy. The business units in turn need infrastructure functionality – purchased from the IT department – in order to achieve their goals. What infrastructure functionality will be realized and at what price are determined though a process of deals between isolated business units and the IT department. Consequently, there is no guarantee for a holistic view on IT or long term strategies.

The investment decisions, derived by maxims or deals, will define what functionality should be realized in services. There is a need for translating functionality needs into actual services that can be implemented. This need, along with the impact of the company’s strategic view, is an unexplored knowledge domain that requires further investigation. The R³-model will be used in this thesis to explain how an organization’s strategic view on IT affects the optimal service abstraction level.

Complexity has been found to be the single most important inhibitor to flexible IT-architecture that would seamlessly transform as the business evolves. SOA is presented as one approach to manage complexity and has by many prominent authors been depicted as such. Properties such as reusability, loose coupling and encapsulation are thoroughly described, but those properties are in fact not new, but rather well-accepted architectural guidelines that have been around at least the last decade. Services on the other hand are new and introduced by SOA. However, how services should be designed is more shrouded in mystery, especially as regards the question of service granularity. The thesis will therefore focus on service granularity. It aims to factorize the notion of granularity and to explicate how the factors are interrelated, in order to aid the design of services that decrease the unnecessary complexity in the IT-architecture. The separation between the different areas of study is seen in Figure 26. The figure also shows that SOA is not the only enterprise architecture venture possible.

![Figure 26 Properties of SOA that affect Enterprise Architecture](image)

Complexity in an IT-architecture originates from six major sources. When the degree of complexity grows the degree of freedom for making strategic business decisions is reduced. IT dependent companies can easily find themselves in a position where the current IT-infrastructure is so entangled with the current way of doing business that it simply is not replaceable, no matter how inflexible it is. The strategic options are then reduced to conducting business either in a revolutionary differently way with a new enterprise architecture, or in exactly the same way; no middle option exists.
11.1 Specified research questions

The path from business strategy to infrastructure scope is quite well understood, as is the path from complexity to business strategy. Service oriented architecture covers the middle ground, and questions regarding how the scope of the infrastructure can be transferred to the scope of individual services and how these, in turn, affect complexity, are bound to arise.

The objective of the thesis is to dissipate the cloud of the unexplored knowledge domain in Figure 25 and to spread some light over these questions. To be more specific, the following specified research questions should be answered in the thesis:

1. How is complexity manifested in Handelsbanken’s IT environment?
2. How does the level of reach for one service affect complexity?
3. How does the level of range for one service affect complexity?
4. How does the level of realm for one service affect complexity?
5. How do the reach, range and realm of one service affect other quality aspects of the enterprise architecture?
6. How are the three dimensions interrelated?
7. Which other aspects of SOA affecting complexity can be found at Handelsbaken?

As seen in Figure 27 these questions will completely demonstrate the property of service granularity. Question 2-4 cover how each factor affect complexity and question 5 cover how they together affect other aspects of enterprise architecture. Question 6, finally, covers how the factors are interdependent. The first question is necessary to know what to study and the last is included in order to illustrate what further effects SOA may have on complexity. The reader should notice that other properties of SOA, affecting aspects other than complexity of enterprise architecture, will not be investigated.

![Figure 27 The domain of research questions one through seven](image)

However, the above questions do not explain how an organization’s strategic view on IT affects the optimal service abstraction level in an SOA. The above presented research questions are therefore extended with specific questions more related to the strategic context of services.
8. What is Handelsbanken’s view on IT?

9. How does Handelsbanken link business strategies to IT operations?

10. How do the reach, range and realm of one individual service depend on the reach, range and realm of the IT-infrastructure?

11. How is the volume of the PARI service dimensioned in relation to boundaries set by Handelsbanken’s IT view and justification of IT investments?

It was in the literature review suggested that a company’s strategic IT view will influence the justification of infrastructure functionality and at the end of the day the ability to cope with and manage complexity. It is therefore essential to establish the IT view of Handelsbanken and the choice of route taken as illustrated in Figure 28 (research question 8 and 9). Furthermore, the impact of the strategic view on the translation of required infrastructure functionality into actual services, as well as the relationship between individual services and the infrastructure has to be examined (research questions 10 and 11).

![Figure 28 The domain of research questions eight through eleven](image)
Investigation

12 IT and business

Information technology is a cornerstone of Handelsbanken’s operations. No matter how important though, IT can never create business, but only enable business practices and work as a foundation for profitability. The critical issue is to make sure that the business organization have the best possible support from IT. One interviewee phrases the problem as: “IT’s role is to give the orderer what he/she requests”. This chapter is devoted to detangling the interviewee’s statement and shed some light over who “the orderer” is and what she/he really wants.

12.1 Strategic view on IT

The researchers had the privilege to meet with the executive vice president of information technology and head of IT department, Mr. A. H. Johansson, to examine Handelsbanken’s strategic view on IT. This section presents the main topics, as laid out by the vice president.

12.1.1 IT’s role in the organization

Information technology has a very central role in Handelsbanken, since virtually all business operations must be completed using some kind of IT support. Costs related to IT represent ¼ of total costs and the head of the IT department have for many years been a member of the executive board. The IT department can be described as the banks production system, or in Mr. Johansson’s terms as “the engine of the car”. The allegory with the car quite accurately capture the general view on IT in Handelsbanken; the engine is important to propel the car, and may in some circumstances make the difference in a sale, but it is in no way the primary focus of the driver. Information technology in Handelsbanken is expected to deliver the power the company needs to do business, but is by no means a driving force in creating new business.

Even though the term IT primarily is associated with hard deliverables such as databases, servers, and web-pages top IT management places a great emphasize on people and processes, especially the co-workers in the IT organization. Business development in Handelsbanken is to a large extent dependent on development of new, product tailored, IT support systems and, therefore relies on 1300 employees of the IT department to be realized. If the IT support system for some reason cannot be completed, the business development effort is also likely to fail.

The dependence on IT support for new product offerings, in combination with increased competition and Handelsbanken’s international expansion, have made the delivery time of IT support a top strategic focus area. If the IT support for new products and processes cannot be launched quickly enough the window of opportunity may close before Handelsbanken is able to enter the market. Although time-to-market for IT solutions is crucial the corporation still demand cost efficiency from the IT organization, in line with Handelsbanken’s overall strategic goals. Top management is aware that other banks in some aspects have a greater IT capability, but doubts that the overall efficiency of these competitors is greater than Handelsbanken’s. At Handelsbanken there is no principal limit for IT costs, but there is awareness that in the end the cost for IT will always be
carried by the bank’s customers and that superb IT capabilities not always translate to customer value.

12.1.2 IT-infrastructure

The justification for IT-infrastructure investments in Handelsbanken is “to create the effective IT delivery”, refereeing back to the discussion about delivery time for IT support. With a well appointed IT-infrastructure less work will have to be done to create IT support systems in the business driven projects, and the time-to-market for new products will be reduced.

The IT department has recently received an extended mandate to take ownership of IT-infrastructure. Historically the infrastructure has been owned by actors on the business side of the organization, which have resulted in a partly fragmented and, in some aspects, incomplete IT-infrastructure.

Allowing the IT department, which have traditionally worked as an internal supplier of services, to take ownership is, however, not entirely unproblematic. The point of having infrastructure that is owned and financed by business units, rather than the IT department, is that business managers better understand the business’ future needs and therefore easier can judge the bottom line impact of infrastructure investments. When beginning to act as owners of IT, a major challenge for IT managers will be to increase their understanding of where the business is heading. Another important aspect is to make sure that the IT-department does not take ownership of systems too close to the top in the IT-pyramid. If a system brings a specific business value, in contrast to infrastructure which brings a general business value, chances are great that the functionality of the system is too specific to be reused in other locations and therefore not effective, or financially motivated, to own the system centrally. Furthermore, measures must be taken to make sure that infrastructure projects do not drain resources and delay business project. The benefits of allowing the IT department to proactively maintain an infrastructure for business systems to be built on are, however, perceived greater than the potential drawbacks of moving responsibility away from business managers.

12.1.3 Challenges for IT at Handelsbanken

The general challenge for the IT department is to become more effective, especially in terms of reduced time-to-market. One of the major technical challenges to this goal is that the bank’s continuous development and refinement of it’s applications over the years have created a diversified and complex IT environment. In addition the IT department must, as discussed earlier, also attain a better general understanding of the business it supports.

One part of simplifying the IT environment and reducing TTM is to charge the IT-department with increased responsibility for the IT-infrastructure, another part is to work with more standard applications. Not every piece of functionality can be home-grown and if good applications are available on the open market they may present a cost efficient alternative to build the functionality in-house. The department is already very apt at integrating third party components with existing systems, but the procedure could probably be further simplified to accommodate for more rapid integration.
According to the vice president there is a general need for simplification in IT. Only if the complete chain of systems, from core infrastructure to end customer interface, are simple to manage and understand can the bank leverage the full potential of IT.

12.1.4 Senior business management and IT management
Just as noted earlier, Mr. Johansson does not view IT as a driver for business development, but rather as an instrument and support for business. In the senior management group his primary strategic role is to critically assess the IT related consequences of proposed business strategies and to, on behalf of the IT department, monitor where the business it heading. In some cases proposed strategies are not practicable due to IT concerns and must be abandoned or postponed; whereas other alterations in business direction can be readily supported by the IT-department without delay. The job of knowing when IT will be a hindrance and when IT will be an enabler to new business is a difficult but a crucial one, especially given the size of IT in Handelsbanken. If business projects are abandoned due to faulty assessment or stall due to overoptimistic calculations big values are at stake.

12.2 Problems and priorities
In discussions with people at various positions in Handelsbanken the authors have identified a number of focus areas for the IT-departments activity. This chapter outlines the major problems and priority trade-offs that IT workers in the company are dealing with.

12.2.1 Internationalization
Having secured a rigid position on the Swedish domestic market Handelsbanken have in later years, together with the other four major full service banks in Sweden, aggressively pursued international expansion. So far the bank has established retail banking operations in the Nordic countries and Great Britain, and is expanding into Eastern Europe. In addition the bank is present on all major stock and investment markets, including China, Singapore, the United States, France and Germany.

The driver behind this expansion is twofold. First of all the bank is, of course, looking for ways to expand its current operations to new markets; to add more revenue streams to its Swedish operations. Secondly, and more important, many customers are international business themselves and demand that the bank serve their needs regardless of geographic location. One interviewee explains her understanding of the problem and its relation to IT:

_The customer wants to see “One bank, One system”. They want the same interface to their business in all countries. But that cannot be solved today, because of the complexity in the IT-systems._

The correctness of the view is verified in the bank’s IT strategy documentation, one of the top priorities create an IT support system with an experience of unity. According to the interviewee business operations have adapted well to the new environment, but IT is lagging behind. Since each country has its own set of laws governing financial operations and, furthermore, has its own way of doing business, the IT-systems need to be specifically tailored to fit each country. The solution so far has been to buy IT-services from local service bureaus, which are familiar with local laws and standards and
therefore can fit a new system both faster and more cost efficiently than the banks central IT-department. This, however, adds an extra level of complexity, since systems in different countries will run on different technical platforms.

In addition, most of the central systems were written to support only the Swedish market and, for instance, lack support of foreign languages. Overall the viewpoint of the central IT-department is very domestic-centric.

One of the architects recall that the IT-department once tried to change the default language of documentation and manuals to English, but had to revert to Swedish shortly after. It turned out that documenting properly in a foreign language was very time consuming, and when employees were rushed by efficiency-conscious project managers the quality of the produced text dropped. Today most of communication, even in those Swedish systems which have an international outreach, is done in Swedish and English is used as sparsely as possible.

12.2.2 Priorities in administration

IT administration at Handelsbanken comprises running and maintaining the many systems. The system owner is a part of the business organization and the key responsible for administration of a single system. Being a part of the business organization, rather than the IT-organization, the system owner does not do any “hands on” work on the system, but is responsible for allocating funds and making decisions about the rate and focus of development efforts related to the system. The system maintainer is the key responsible for maintaining and developing the system. When a new requirement arises the system owner will allocate a budget and order implementation of the requirement from the system maintainer, who will in turn set up a project team and carry out the needed development. The actual running of the system, i.e. installing, executing and monitoring the software on the mainframe, is done by the IT departments operations group. System owners are debited based on the amount of system resources used, with the amount of storage and the number of low level instructions executed by the microprocessors in the production environment as a basis for cost breakdown.
The different systems are created by development projects, which are in turn primarily initiated in response to specific business needs. The project, led by a project manager, operates on its own budget and is generally responsible for developing one or more systems to support the desired business functionality. A major part of the work is to make the newly developed system work in the environment of already running systems and enable them to exchange data with other systems where required. Creating these, so called, relations between two systems usually mean coding separate communication modules in one or both systems. For example a system responsible for transferring money, will among other things, need a relation to the subsidiary ledger system, that holds balance for customer accounts, and another relation to the part system, that hold personal data regarding the customers. Over the time the number of relations tends to increase and there are systems with well above 500 relations currently in operation. Development is however not the only cause for new relations between systems, special circumstances such as mergers and acquisitions tend to disruptively add extra relations in order to seamlessly pass information in the architecture.

The system owner is, as the administrator of a system, hence not only responsible for the systems functionality, but also for making this functionality available to an ever increasing number of client systems. The crucial thing to realize here is that the more relations the more complexity; just imagine the process of receiving 500 approvals from other system owners before implementing a change and afterwards making sure that all 500 systems work as expected.

There is also a budget conflict between systems being developed and the systems they are integrated with. In general the project manager wants as much of the integration related logic as possible to reside in the existing system, since the expenditure for developing this logic will then end up in the system owners budget and not in the

Figure 29 Professional categories
project’s, whereas the system owner ideally would want a single set of logic to handle all relations; it is often the case that projects running low on budget, requests a lot of new functionality to be implemented in already existing systems.

Another dilemma that system owners face is the tradeoff between development costs and administration costs. In software development there are often opportunities to make “quick and dirty” solutions that reduce the time and cost for development, but which may result in higher maintenance costs over time. A majority of the interviewees takes the view that developing systems that are easy to maintain must have priority over the initial cost. One system owner puts it very pragmatically:

> Once the development cost has been taken, it will soon be forgotten. The administration cost, on the other hand, will remain and make itself reminded each year.

A system owner at Handelsbanken Capital Markets, argues that system maintainers and system owners have to take a greater responsibility for the long-term view and speak up about the consequences that requested business functionality will have on maintainability and system robustness. The point is, that even though business managers may not understand how IT works, they are well apt at judging the extra business cost generated by unreliable IT-systems and therefore prepared to put money on the table in return for some extra long-sightedness if only the facts regarding system robustness is communicated by owners.

The level of understanding IT and IT-related questions does, however, vary greatly among business managers, often making it difficult to discuss and explain IT-investments. A system orderer comments:

> Those who have participated in many development projects naturally have a greater understanding than others, but in many of the regional head offices a view of how IT and IT-development works is completely lacking…it is often hard to communicate why IT development is so time consuming and expensive.

### 12.2.3 Priorities in development

Development projects in Handelsbanken are in general not very unlike development projects anywhere else; in the end it comes down to the variables of time, cost and functionality, with different approaches to maximizing functionality while keeping development time and cost at a minimum.

In Handelsbanken most development has traditionally been done in-house, but purchased generic off-the-shelf solutions are gaining ground and will, in accordance with the most recent strategies, continue to do so. The general idea is that purchased software is much more time and cost efficient, since several customers will “share” the vendors development cost and the vendor in general already have a product with the sought for functionality. Opponents argue that the majority of initial costs are related to integration with existing systems and, hence, that the vendor price only makes up for a small portion of the total time and cost to get the system up and running. Furthermore several IT-workers have complained that business managers tend to oversee potential problems with packaged solutions, in order to gain perceived cost and time benefits of the product. One of the IT architects explains:
Not seldom are commercial alternatives compared to internally developed ones, but the general requirements that the internal development organization will have to adhere to are often overseen when reviewing the commercial alternatives. One compares, so to say, apples to pears and concludes that the bought systems are better.

The general requirements overseen are often linked to long-term maintainability and designed as rules of best practice to make the system stable and easy to maintain. Business managers tend to favor relaxing these requirements, instead of going through the lengthier and more expensive process of ordering a custom made system.

### 12.2.4 Consolidation of functionality

Another approach to create more functionality for less money is to build general components. A general component is a system that collects functionality common to many systems in one, with the benefits of creating a coherent view of all data and reducing the number of relations each system need to have. The system of this study, PARI, is the first component of this type the Handelsbanken has put into action. The idea with PARI, as previously explained, is to consolidate all information about parties involved in the company’s business transactions, in order to provide a single point of integration for all systems using data about any party and make sure that this data is coherent and up to date. Furthermore PARI provides a generic lookup service, enabling using system to connect and very precisely specify and retrieve relevant data without any customization on the PARI-side.

Contemplating the fact that there are close to one hundred systems that gather and store some kind of party data and that the traditional approach to integration is to program a custom made lookup service for each consumer, the idea of a single consolidated view with a generic lookup service seems well thought. There are, however, a number of problems related to this approach.

One of the major obstacles is the problem to systematically generalize to a level where services become reusable. Developers are used to implement solutions to specific and well defined problems. To construct a solution which handles both present situations and potential future needs efficiently requires a new set of insights and a more comprehensive understanding of the business.

Building general components also require a commitment to the component on the business level. Managers from different business units will have to agree on a common set on requirements on the service provided by the general component and changes will need to gain approval in the group. The result is often that managers emphasize the uniqueness and importance of their own set of information to advocate against a common component. This may seem irrational but, as one system owner point out, the division into business units does exist for a reason. The kind of business each unit is responsible for differ a lot and therefore the demands on IT-support will also differ a lot. Consequently a consolidation of IT functionality to reduce complexity may cause even greater complexity on the business side.

General components do, however, on a purely operational level, possess appealing qualities. One is that personnel operations become more efficient when there are a fewer number of systems to keep track of. Today the sheer amount of systems in many units
results in a situation where all personnel has moderate knowledge about a handful of systems, but no one has expert knowledge on any single system.

13 Handelsbanken’s IT environment

Banks, with their enormous need to accurately process vast amounts of information, were among the first business to employ computers, and Handelsbanken was no exception. The bank has seen every invention and trend in IT over the last 40 years and, over time, developed a respectable amount of systems itself. This section will give a brief overview of some of the central technological concepts currently employed.

13.1 History of systems and architecture

This chapter will provide insights on three topics important to readers general understanding of the IT environment. First the technology foundation, in terms of computers, databases and operating systems, is discussed. After that the internally developed “Service Concept”, which bears a strong resemblance to SOA, is introduced and finally the principles governing investments in IT-architecture at Handelsbanken is discussed.

13.1.1 Foundations

The bulk of all business applications in Handelsbanken are, and have always been, running on an IBM mainframe. The mainframe is essentially a huge computer optimized to perform a large number of small business transactions (such as withdrawals, stock purchases etc) simultaneously and at high speed. The core of the mainframe applications are the z/OS operating system, the DB2 relational database, the IMS transaction process manager and the MQ delivery manager, all purchased from IBM.

Business applications running on the mainframe are to a great extent developed in the COBOL programming language. In addition to the mainframe based applications there are also a number of business applications running on Windows, Unix, Open VMS and I-series. The user interface is separated from the application logic and is based on dialogues displayed in a web browser. The dialogues are primarily written in the Java-based Espresso framework, which is running on an IBM Websphere Application Server. This solution have complemented an older terminal based solution.

The mainframe based environment is perceived as rigid, efficient and thoroughly tested. The IMS-transaction system, for instance, has its roots in 1960:s Apollo space project and the currently used version of the Cobol programming language dates back to 1985. But the rigidity is problematic from an integration perspective. One example is that no IMS-transaction can be allowed to access a non z/OS system to retrieve data whilst processing, because the IMS system, like any other synchronous transaction system, will wait for a response for every request it sends to an external or internal party. The problem is that with the large transaction volumes of Handelsbanken’s system, thousands of stalled requests will soon accumulate and will decrease the performance of the system and even cause failure. In worst case a failure in the transaction processing manager will lock one or more tables in the database and thereby cause other applications to fail, which of course is not only an IMS specific problem. The Cobol programming language also have a number of inherited limitations, for instance variable
names can only consist of capital letters and modules cannot call themselves in a recursive fashion. Programmers are, never the less, enthusiastic:

“The environment is fixed, so you can concentrate on solving the problem.”

According to the interviewee Cobol imposes so much structure on the programmer that there is often only one way to solve a given problem, in comparison to the various coding “styles” and “schools” that has grown in, for instance, the object oriented community. The default structure makes the code easier to understand and maintain once written, but also helps to standardize and automatically generate. In fact up to 90% of all code can be automatically generated, leaving the programmer only with the small portion of logic that is unique to the application he/she is currently writing.

13.1.2 The Service Concept

An approach to standardize how information is exchanged between two business systems and between dialogue and business system is the so called “Service Concept”. Service providers provide a number of general services, each implementing some functionality, which service consumers can call with a predefined set of parameters. The communication is message based and as response for each request the consumer receives one or more predefined messages with data. Each response message is tagged with its type and is dynamically generated based on the request.

In a very simplified example of a typical message exchange between a dialogue and a subsidiary ledger system the dialogue starts by sending a message containing the name of a customer to the ledger’s account lookup service. The ledger system then responds with one message for each account the customer has in the bank. In this example the dialogue is only going to display information about checking accounts and, hence, scans the message type of all returned messages to identify checking accounts. All savings account messages are simply ignored and discarded. The approach of having broad general services have the disadvantage that service consumers need to handle lots of irrelevant data, but also the advantage that the service at any time can be further extended with new types of messages, since consumers will only pay attention to those messages they are already familiar with.

The core set of ideas in the Service Concept very much resembles the ideas that constitute Service-Oriented Architecture. One of the major differences is that SOA proponents strongly argue for XML-based messages transmitted using web-service technology, whereas Handelsbanken has implemented the Service Concept by documents based on a Cobol data structure, transferred over MQ. The main reason for this implementation is that Cobol, IMS and MQ are stable and widely deployed technologies which have worked for years and years, whereas web-services were only in its cradle at the time Handelsbanken started developing its Service Concept. It should, however, be noted that web service interfaces are becoming increasingly popular among vendors of commercial banking systems, and that Handelsbanken’s recently expressed IT strategy recommends using a high degree of general standards.

13.1.3 Architecture

The Service Concept is an architectural infrastructure innovation. The problem with architecture is, as our theory foundation shows, that it usually cannot be justified with a return on investment calculation, since factors determining pay-off are too uncertain.
The approach to justifying investments in architecture and infrastructure in Handelsbanken is generally to bundle them with some major business application. One application architect states:

*All investments in architecture have been driven by big business projects.*

The interviewee continues to explain that the problem is that the usefulness of the infrastructure investments are seldom or never brought to business managers attention, and hence they cannot value it in terms of business lost or gained. This fact makes it problematic to discuss and raise funds to infrastructure projects in a proactive manner.

### 13.2 Handelsbanken’s Data Model

Handelsbanken’s Data Model, HBDM, is an attempt to standardize information throughout the corporation. It is originally an adaptation of an IBM product called the Financial Services Data Model. Since the use of a common data model is integral in both SOA and the systems studied at Handelsbanken, this section will be devoted to an evaluation of HBDM.

#### 13.2.1 Function

A data model describes and structures data in a predefined way, with the goal of achieving a better understanding of the data itself. The model structures and describes information in a hierarchical way based on the entity the information describes. An entity at the top level, in HBDM, can be classified as any of nine topics, such as Involved Party, Arrangement, Geographical Location or Resource Item. On the second level a party can be an organization, an organizational unit, an individual, or employment position. An individual can in turn have subtypes, such as gender, occupation and religion. In addition all parties have a number of descriptive attributes such as name and role in the relation to Handelsbanken. In this way any information regarding the banks business can be precisely classified.
The benefit of having a data model is that data can be handled without any detailed understanding of its meaning. An arrangement will for instance always have a title and one or more parties signing it, regardless of which country, industry or organization the agreement was created in. If a computer system works with “title” and “signing party”, rather than for instance “lending agreement” and “customer signature”, it will be able to process any agreement, without any modification to the system. The other advantage is that the model standardizes the meaning of data fields to avoid unnecessary confusion. For instance the term “delivery date” may indicate the date the product was sent to a customer in one place and the date it was received by the customer in another place; the data model standardize one interpretation.

13.2.2 HBOM – an object representation of the data model

HBDM is a conceptual model only and therefore has a limited use. Handelsbanken’s Object Model, on the other hand, is an object representation of HBDM that can be used as a template for creating data objects that can be used when building systems. What HBOM essentially does is that it collects the different attributes associated with an entity in one place, the object. Essentially the object is a list with attributes in one column and values in the next, where the attributes are derived from HBDM.

13.2.3 Usage problems

Although the idea of a common data model have many benefits, it also inflict a more abstract relation between the developer and the data. The problem of translating data in one system to data in another has become a problem of translating data in one system to
fit the data model. The primary studied system, PARI, is for example a central component which collects, stores and retrieves data about involved parties, i.e. customers, branch offices and employees, without much consideration of the meaning of the represented data. PARI can operate this way since it does not include any business logic, but in the service consumer systems the knowledge of what the data really stands for is often crucial.

Traditionally all service interfaces have been custom made to fit the needs of the consumer and the consumer have been able to in detail specify exactly which data is needed and how it should be delivered. The work of picking out the relevant information has been a job for someone in the service provider organization, but with the introduction of the Service Concept and HBDM, selecting data has now become a concern of consumer side programmers. Services, in order to be generally usable, now sends an abundance of data, modeled as HBOM objects, to their consumers. In practice this means that the people responsible for implementing the consumers will have to gain a throughout understanding of the entire data structure, since they are now responsible for picking out the relevant data. If the system is also responsible for delivering the result on HBOM-format, developers will, moreover, have to understand how to translate the attributes they are working with internally to fit in the HBOM model.

The result is what one interviewee described as “messier projects” with an increase in time consumption whenever HBOM is used. Project managers routinely add extra weeks of development time to accommodate for HBOM amendments. Several of the interviewees, however, describe a learning curve. Working with HBOM was initially perceived as very cumbersome, but after having learned to navigate the model and understand the name standard most of the developers felt more confident.

The problem has several roots. First of all HBOM is a quite comprehensive model. There are sixteen objects, with about 15-30 attributes each, only for Involved Party. In other words, there is well over 300 attributes just to describe a person. The second problem is that all abbreviations are in English, a language not very well understood in the organization, and that a limitation in Cobol restricts the length of attribute and object names to 16 characters in capital letters only, hence all names are English abbreviations. For instance FNCLLGLSTTP translates to Financial Legal Status Type; a field indicating whether or not the party is bankrupt. The abbreviations are standardized, so that for instance LGL is always short for Leagal, but there are glitches where other shorts are used to mean the same thing.

Another aspect of handling the model is how the data is received in the consumer and problems related to manage the redundant information received. These problems express themselves differently depending on which platform the developer is using. On the Cobol platform the problems are related to the fact that messages are generated dynamically. In programming in Cobol, which is primarily designed to handle static structures, complexity increases significantly when the programmer have to account for reception of an unknown number of messages, much due to the way Cobol allocates memory. In Java, on the other hand, the prescribed technique is to collect all data into a tree structure, referred to as a Document Object Model or DOM. Due to the way HBOM is designed the tree structure often get so deep that it is inconvenient to handle for the programmer.

A partial remedy would be to engage data experts in early stages of development, to specify exactly which objects and attributes that are relevant. Today this task is often
left up to the programmer, who should not be expected to possess in-depth knowledge of the data.

13.2.4 Structural problems

In addition to the practical problems associated with using HBDM/HBOM there are also questions of how applicable the model is in different parts of the enterprise. IT staff at Handelsbanken Capital Markets argues that although HBDM is sometimes used, it is not fully applicable to their investment banking business, since the model was originally developed for retail banking. One example of this is when a project dealing with financial instruments did a HBDM mapping of their data and discovered that the HBDM term for “instrument” was Financial Markets Offering. The team members soon started using the term “FMO”, instead of the generally accepted “instrument”, creating major confusion in communication with business representatives and other project teams.

One architect at Handelsbanken Capital Markets argues his personal opinion that the bank is over-confident in believing that one general data model will be enough to create a semantic bridge between all systems:

*It will be so expensive and hard to adjust every single system to fit such generic model, that it simply won’t be worth it.*

He proposes a more agile style, where data models are created as needed and are then refined and reused in other projects with similar demands. The key is to create only what is needed and that the created model must be understandable by both the orderer and the developer. The problem with ad-hoc models of this kind is that they over time tend to become very fragmented, exactly the problem the general data model was designed to overcome.

14 Systems and projects

Handelsbanken’s IT department accounts for more than 500 systems. This chapter will describe the PARI system, which is the focus system of the study, as well as a number of related systems. It will also describe the IT-organization at Handelsbanken Capital Markets, to provide a better overview of how different parts of Handelsbanken are interacting.

14.1 PARI

PARI is, as already explained, the company’s central component for handling parties. The primary reason for creating the component was a new EU-law, generally referred to as BASEL II, which stipulated that that creditors must rate and assign a credit limit to all their customers and that a customer must only have one rating, regardless of how many countries the creditor had relations to the customer in. The practical implication of the law is that banks must now have a central register where all customers are listed, regardless of nationality, in order to do a correct and coherent rating.

Even though BASEL II was the main driver behind PARI there were also other important aspects to consider. One of the more important was to make it easier and less expensive to develop internationalized applications. Previously customer data for each country where the bank had activities were collected in one or more national databases and developers would have to build separate relations to each and every one of them.
Now PARI takes care of the integration and offers a master view of the most current data. In addition there were several Swedish databases holding different kinds of party data for different data consumers within the national organization. It was also assumed that maintenance costs can be greatly reduced by consolidating this data into one single database accessed through PARI. Because of legacy systems accessing the old data sources they will, however, be operational for a foreseeable future, and no cost savings will yet have to be proved.

In order to facilitate the consolidation of data sources all newly built or upgraded systems are expected to only access PARI. Already existing systems may however still use legacy data sources and, furthermore, due to the fact that PARI is not yet fully functional new users of old data sources are still being added. One problem is that PARI does not yet handle corporate structures, such as divisions into subsidiaries, which is needed to handle large clients correctly for corporate banking and insurance handling. Another problem is that the Service Concept requires consumers to log in before using a service, something that especially purchased systems have difficulties with, since they often have their own mechanisms for user management.

### 14.1.1 Functional description

PARI is constructed as a front for a number of other data services, that is PARI does not itself hold the data but instead calls the appropriate service and forward the result when receiving a request. The most important of the underlying services is the POOL-IP system, which holds the majority of the bank's customers as well as other data. POOL-IP extends its predecessor, POOL, and is designed to hold all party data when the other legacy systems are wound up.

Responses from PARI are received according to the HBOM format. The apparent advantage is that it allows for extending the underlying databases with more information without any need to rebuild consumer applications; new data will be sent in new messages that old consumers will just disregard.

Internally PARI is constructed in a modular fashion with one module for each data source, making it very easy to extend, both to add new data sources and new services for consumers to use. However, some performance critical operations, such as end of year books closing, are not yet available in PARI, but must be accessed from POOL directly.

### 14.1.2 Accessing PARI

One of the major concerns with the PARI system is the difficulty for consumer programmers to correctly access the system. The interviews generally paint a picture of their first encounter with the system being very stressful and all agree that it takes longer to get started using PARI than other systems. The general attitude to the concept of one consolidated data source is, however, positive. Most programmers seem to like the idea of having party data from all business units in one place. The problem is that the vastness and structure of data makes it hard to handle. A Cobol programmer estimates that:

> A first time user probably needs 2-3 times as many hours to write a PARI connection than a connection to a traditional system.
To minimize the impact of using PARI on the development process projects seem to take one of two approaches. They either assign one programmer to master PARI and write a separate module that does all the communication and “translates” the result to something that the other programmers can use, or they write one instance and use cut and paste techniques to save time on subsequent usages.

### 14.1.3 Data selection

One of the downsides with consolidating several databases into one is that the information often becomes difficult to grasp. Earlier the bank used different systems for different types and different users of data. Once the correct database was accessed the semantics of the data were often quite clear. In PARI information about widely different logical entities, such as English customers and Swedish branch offices, dwell side by side and may be returned by the same query. In addition the use of HBOM and the Service Concept has transferred the responsibility for selecting meaningful data from the vastness of information from the producer to the consumer. Two main patterns, contingent of the platform the consumer uses, of data selection difficulties have been observed.

#### 14.1.3.1 Cobol and selection keys

PARI’s query interface defines three keys service consumer use to describe which data they want to extract. The first key holds a number of fields which specifies what entity to lookup, for example an organizational number and an indicator of which country it applies to. In the second key the programmer can specify how it wants PARI to carry out the request, for instance by only searching in Swedish databases. The third key specifies which type of objects to return. Example of limitation based on the third key may be to only return electronic addresses that are telephone numbers.

The system with search keys has received massive critique and developers find that specifying the keys correctly is a major challenge. One problem is that many entries can be done in all three of the keys and the programmer will need a good view of both how the business works and how PARI works internally to understand where the specification should be made. Imagine that one is searching for a Swedish customer’s official address. There is one language specification in each of the three keys. If “Sweden” is specified in the “what entry” key the interpretation is that the search key used, for instance a personal number, is Swedish; but this search may still return an Norwegian address, if the Swedish customer also has address in Norway. If one specifies “Sweden” in the “how to execute” field, PARI will only search Swedish registers, but the Norwegian address may still be present in a Swedish register. If then, finally, the country key is specified in the “what to return” field the address returned will be Swedish, but if the other keys are not correct one may get a response for a Norwegian customer with a Swedish address. One frustrated programmer comments:

> PARI is immense and it is hard to gain full knowledge of what it can deliver. Everything is based on that one sets the correct keys, but it is hard to know whether the call has been done correctly. ...One doesn’t necessarily know how it works on the business side, that you can be a customer in Sweden, Norway and Finland at the same time.

The fact that you do not necessarily notice that one key is missing or set incorrectly is really bothering programmers, since it makes testing and verification of the system...
much harder. Programmers feel that they do no longer have control over their product; the system may work well for years and then one day deliver a faulty result because of an erroneous search key. It should however be pointed out that the three key system is a very powerful tool, where the programmer are given the power to in tiniest detail specify which data to retrieve. Given the vast amount of different data contained in PARI, powerful search tools are needed.

Except from using a demanding search interface PARI also demands that the programmer programmatically review the retrieved data before using it. If the party, for instance, have a confidential address this will be marked in a separate field, thus the programmer have to evaluate the status of this field and based what it says decide if the user should be allowed to se the address or not.

One have to know the meaning of a number being a customer number and not a reference number and so on. ... The bottom line is that if you, as a developer, make a mistake about the interpretation of a field the system will not behave as expected. Earlier the responsibility for getting the semantics right were the designer's, who agreed with service suppliers to deliver a flat structure containing exactly the data needed for the development. No developer would then have to learn the data structure of the system, only the fields that were needed.

Earlier, when using custom made relations, the service producer could be programmed not to retrieve any confidential data if the receiving system were not going to use it anyway. Now the client programmer will have to learn to identify where special attributes that needs to be inspected are located and do the processing on the client side. This may seem like a minor problem but never the less adds to development time and error frequency, since programmers in many cases must now learn more about the properties of the data than earlier.

14.1.3.2 Java and information overload

Whereas Cobol programmers experience that they get a precise and manageable number of objects to handle once they master the search keys, Espresso programmers have the opposite problem. The Java based Espresso platform is used to display the web pages that customers and branch offices personnel use to enter information. In order to save programmers the trouble of composing the search keys, a set of reusable Java classes was developed to make the call to PARI and format the result in a way that is readily accessible to Java developers. The Java routine, however, only allows for a limited specification of the search, equivalent to the “what” key in PARI. When executing the Java-routine calls appropriate services in PARI, collects all the response messages and builds a tree structure, usually referred to as a Document Object Model. The DOM can then be traversed by the developer to pick out appropriate values. With the amount of data represented in PARI, the lack of possibility to further specify the search and the general structure of HBOM, the resulting tree often gets very big and difficult to navigate. There is an information overload on the receiving side.

Some measures have been taken to limit the consequences, for instance a new feature in PARI allows for a limited amount of 20 addresses to be retrieved at one. This is practical since large enterprises can have hundreds of addresses for different branches all over the world.
14.1.4 Documentation

One way to reduce difficulties involved in using advanced technical systems in general, is to make an easy to understand and comprehensive documentation. The maintainers of the PARI system have taken this task seriously and created several manuals governing different aspects of using the systems, aimed at different users.

There are three different specifications written for business architects describing the HBOM message structure and the names and intended usage of different fields. One specification is for Handelsbanken’s organizational units, the second one is for customers of foreign branches and the last one describes domestic customers. The reason for keeping three separate specifications is that relevant attributes and the semantic meaning of the attributes have been found to differ greatly between the tree sets of data. In addition to the data specifications there is also a rather advanced technical manual describing how to make calls to PARI and how returned data is structured.

Even though manuals are written in with the best intentions several critical remarks have been made that the specifications and manuals are difficult to understand for first time users. Several programmers suggest adding code examples to aid those unfamiliar with using the Service Concept, search keys and HBOM. Developers also need an understanding of the actual data and context of the service. Suggestions for more documentation targeted for other professionals such as product owners, business analysts and system owners have also been presented.

There are also missing administrative routines around the distribution of documentation. Several interviewees state that they either had difficulties retrieving or did not even know about the PARI documentation. The system maintainer in response states that she expects all potential users to contact her in order to administer manuals and provide support on how to use the system. One interviewee expresses her despair:

    It is unreasonable that everyone has to attain meetings in order to orally take part of the information.

She continuous to state that written documentation, for different target groups, is a prerequisite in order to make the business side and the IT side to understand one another.

14.1.5 International relations

PARI will, according to plan, continue to work as a common front end for national databases. POOL-IP, the main Swedish database, was never intended to encompass data from other nations, but only to be one of several national databases. Currently, however, a technical issue makes it impossible to have online data from other countries and consequently a copy of all international data is imported into PARI/POOL. The import runs overnight, so that the information is available in PARI the next day.

Although PARI contains data for the whole enterprise, it is owned only by the Swedish organization. This implies that functional demands are specified from a Swedish point of view, something that often get consequences for the possibility to use the systems by international units. One example is that Swedish information always has precedence, that is if one piece of information conflicts between two countries the Swedish information is automatically deemed as most correct. In other instances it is more
problematic to judge which piece of information is the more correct one, if updates from several countries arrive.

The problem with online data causes other problems, for instance that a foreign office cannot make transactions in PARI dependent systems until the next day after a new customer is added. This puts a practical limit on the possibility to utilize the central corporate systems, instead of having local ones, since most customers does not accept that they will have to come back the next day to finish opening their account.

The PARI user interface, the dialogues, is also developed for Swedish use only and the main clientele are the Swedish retail banking organization and Handelsbanken Capital Markets. The reason for using the dialogues only in Sweden is partly due to some functionality in the dialogues that retrieve information from official registers, but also due to the fact that international customers are automatically updated from local registers.

14.1.6 Performance
The scope of PARI is considerably larger than previous any database, something that has affected performance in a negative way. The relational database underlying PARI/POOL-IP contains a total of 8 tables, each containing millions of lines, in contrast to 2 tables for the old POOL system. A response is compiled by combining data from the different tables according to certain rules, which is derived from the search keys. Compiling data from eight tables takes considerable longer than compiling it from two tables and the consequences have been measurable, especially in lookup intensive applications like the online banking system.

14.1.7 Service as a part of a process
There are endless occasions when party information is needed in the business activities. Hence, an infinite number of systems use the information provided by PARI, either direct from PARI via a service, or through the use of POOL. There is also numerous ways through different systems to enter or update the information stored and there exist no general guidelines or principles on how to enter the information, resulting in inconsistent and unreliable information. For instance, there is no coherence on how party information is displayed in dialogues or how systems should format an entered attribute in order to pass it on to POOL/PARI. A concrete example concerning postal addresses from the PARI documentation highlights the predicament.

PARI has structured the address according to HBDM in many different fields such as: Postal Distribution Address, Post Code, Postal District, Postal Address Country, and Postal Address Country Code. This allows an address, in theory, to always be correctly formatted. However, it requires all systems using PARI to figure out how they should be implemented in order to store an address correctly. If for example a Swedish dialogue does not show the field Postal Address Country Code but all others, how should one enter a Swedish postal code, “33380” or “SE-33380”? Most users in Sweden would probably use the first version but how would one enter a French address, “33380” or “FR-33380”? Making the field available could of course eliminate the problem. However, is it obvious for French users to enter “FR” in Postal Code for French addresses? The dilemma gets even trickier if the address contains state or providence, where should that be entered, as “VA 33380”, “33380 VA” or just “33380”?
The problem is not the construction of PARI or HBDM, but the usage thereof. One business analyst articulates the problem as: "Shit in, shit out" or in other words, the chain is not stronger than its weakest link. The allegory of a chain highlights the concept that the PARI service is a part of processes, but often just a small part. Party information is seldom the main artifact of a process but rather treated as a prerequisite and should therefore be easily accessed for the service consumers. The interviewee expresses her view that PARI has to take a wider perspective on the Party Information Handling Process and not just store and provide party information without regards of the business processes involved. The responsibility has to be end-to-end, meaning from that someone, machine or human, receives party information (electronically, orally, or written) to the presentation of the information (in dialogues, over batch files, or on printed lists). The Party Information Handling Process does not start with a system entry in PARI and does not end when PARI presents the information to another system.

14.1.8 Adjusted service granularity

Using the R³-model the researchers constructed a proposal of more finely-grained services and asked a number of interviewees for their professional opinion of the new services. The proposal was based around an adjustment of the realm dimension to divide the party concept into finer grained units, such as “Handelsbanken organizational unit”, “business customer”, “private customer” etc. The aim was to reduce the amount of conflicting information in responses from PARI and, in turn, the difficulties for programmers and others to grasp the data. The general responses from the developers were positive, one Java developer explained:

If one were to make modifications to services, one should make them smaller. The interface would then be easier to understand and it would work to reduce the depth of the DOM-tree. Thereby a better understanding of where information is stored and how to retrieve it would be gained.

PARI specialists, however, raised a warning flag. Dividing PARI according to semantic differences between party data would require developers to know which kind of party they requested. One of the key advantages with using the party notion is that other systems can refer to parties involved in a transaction without discrimination. A subsidiary ledger system, for example does not need to know whether the owner of a savings account is a commercial or private customer, as long as the account belongs to someone. Splitting PARI into a number of services with a more limited realm would hence only add to the programmers’ workload, since information regarding type of party would have to be known to carry out the request.

The second objective for creating a party service with a smaller realm was to simplify the work of the business analysts. It is estimated that of the total time spent on integration more than half is comprised of business analysis, in this case to analyze which fields in HBDM and PARI that corresponds to the data required by the other system. A business architect explained her view of the problem like this:

Advantage can be gained in either of two ways when reducing the level of abstraction. One way is to chip off redundant data or data with different semantic meaning, so that these entries don't have to be investigated. ... The second way is if services could be specified in a manner that reduced the number of controls that have to be made.
The first option can be exemplified in PARI, which stores both data about private and commercial customers, as well as information about employees and organizational units of Handelsbanken. The data held for the organizational units differ a whole deal from data about customers and the occasions where a system is indifferent between branch offices and their customers are few. Hence, separate services could be employed for customers and organizational units.

The second option affects the range of a service, rather than its realm. The checks one wants to avoid is the above mentioned inspections of special attributes that the programmers have to do. By creating services that explicitly left out certain data the programmers would have to know less about the data they handled and analysts would have a smaller problem of transferring this knowledge. One example could be a service that does not return matches for deceased customers or addresses that have been marked as unlisted.

14.2 Systems at Handelsbanken Capital Markets

Handelsbanken Capital Markets, HCM, are responsible for the banks operations on the world’s capital markets. The unit operates as a strategic business unit and as such operates its own IT-administration. The IT staff consists of approximately 270 persons, together responsible for about 50 IT-systems.

HCM employs a wide variety of platforms and systems. Development of business system is primarily done in Java and .NET on UNIX and Windows platforms, but some functions run on the VMS platform and, in addition, integration with systems on the central mainframe is almost always needed. The result is that processes generally stretch over numerous systems on different platforms and overlapping functionality between systems is a common occurrence. Systems in the environment are tightly coupled, in part due to the development culture but also due to lacking separation between internal data models and external ones. In at least one case an internal data model, from a purchased system, have been directly exposed and used as a data model for integration, with fatal consequences during a later update cycle when the vendor decided to change the systems internal model.

Another consequence of running systems on disparate platforms is that real time interaction is hard to achieve. In cases where the immediate passing of information is impossible a semi-manual routine is used. In order to facilitate easier integration HCM are currently developing an IT-strategy based on the notion of components and has also started tests with web service and SOA. One finding related to this is that web services tend to degrade in speed when services are built to finely-grained, because of the overhead involved in web services.

14.2.1 Derivative systems

A deriveate is a financial instrument which value is directly dependent on the price of an underlying security. Derivatives usually exist in two shapes; as vendor specific so called structured derivatives and as standardized derivatives, common to many vendors. Structured derivatives are continuously being developed and introduced on the market, but only a fraction of them become successful. Successful products are generally copied and sold by other companies and as that happens products turn into standardized derivatives. The diffusion cycle ranges somewhere from one to six months and time to
market for new product ideas are therefore in general very important, since the prize pressure gets immense as soon as other actors enter the market.

The fast diffusion rate for derivatives put unique constraints on development of new systems. On the one hand the bank wants a rigid IT support to be able to handle large quantities of transactions throughout the product life cycle, but on the other hand it does not want to invest in IT-support for all those products that are introduced but that fail within the first six months. The problem is usually solved with a combination of automation and manual handling, but the latter is error prone and implies a lot of risk.

An ideal support system for derivatives would require manual intervention only when some kind of error had occurred. Given the amount of manual steps in the process today there is a great savings potential, but more important is that it allows the bank to do more business with less risk than if manual handling is used.

In general three factors are taken into account when discussing IT-systems at Handelsbanken Capital Markets. The first is stability, that is how many interruptions caused by errors the system generates. Secondly flexibility, in terms of the systems ability to handle products introduced after the system was delivered. The third evaluation criteria is the amount of automation the system offers, where more automation implies fewer points of human interaction in the business process. The three factors rated differently for different types of systems, depending on the type and amount of business transactions they handle. The current derivative handling systems are deemed as stable enough, but lack in flexibility.

### 14.2.2 Stock systems

The banks stock systems handle the purchase and sales of securities on stock markets. The number of transactions in these systems is very large, especially since the banks customers can access the system through their internet banking connection and manage their own stock portfolio. Furthermore, securities transactions are time sensitive, since the price of the security can fluctuate on a per second basis. Due to these facts stability is ranked as the top priority, given that even a short disturbance may result in major losses for customers.

Stock systems face different integration scenarios than the derivatives systems. One quite common scenario is that large corporate customers asks the bank to make their stock available for trading on marketplaces that the bank is not yet connected to. In order to efficiently handle transactions on the new marketplace integration between the bank’s trading system and the marketplaces’ trading system must be created.

Another typical problem area for stock systems is related to reach. Competition from online stock brokers has made customers aware of other options then The Nordic Exchange. These markets have, however, traditionally been managed by regional branches on separate systems and real time integration has therefore in some instances been insufficient. One solution that has been used to temporarily solve problem is to let the customer do the transaction in the online system and then employ a stock broker who manually enters the transaction in the local system. Naturally, it has been a top priority of the bank to increase the reach of these systems.
14.3 Currency transfer

Handelsbanken transfer currency, on behalf of thief customers, to other banks using SWIFT which is an organization specialized in inter-bank money transfers. SWIFT provides a communication hub which the bank’s system can connect to and make payments to other banks. The communication on Handelsbanken’s side is managed by a purchased software, that have been integrated with the banks other systems.

Due to the international status of the system it is organizationally owned by the international branch of Handelsbanken Market. Foreign regional headquarters could potentially get their own currency transfer systems, if they perceive that they are given too little influence over development of the central system.

The top priority for the system is stability; nearly 4 billion SEK are transferred through the system each day. But the availability demand and the reach of the system puts stress on the supportive organizations. Because of the fact that this single system supports branches in so many time zones, maintenance can only be allowed during very restricted hours. All work that implies taking the system off-line, for instance updates, must be performed in a narrow time window between Saturday noon and Sunday noon, in order to not affect the business anywhere in the world.

Automation is also a priority. There is currently a lack of automated support for certain rare kinds of transactions, all together these transactions, however, occupies two full time personnel in the central department. The product in itself is very stable though, so flexibility in the sense used earlier is high in priority.

Improvements in the SWIFT system have similar business benefits as improvements in the derivative systems. Removing manual handling would, thus, improve both cost efficiency and quality. This in itself would of course be positive, but the big pay-off would be realized through handling even more specialized transactions for customers, without too much risk involved.

14.4 Cash-Pool

A cash-pool is an advanced cash management service for large corporations with subsidiaries in several countries. The idea is to connect the customer’s accounts to one system and provide a single view of the total corporate liquidity, regardless of currency and nationality of each subsidiary’s accounts.

Handelsbanken’s has a cash-pool solution that is currently under development. Due to the amount of offered functionality the development has been very difficult. Essentially the project team must integrate the new system with each and every existing system that in some way handles cash for customers. Since all the international regional headquarters over the years have bought their own systems, instead of using central ones, the number of integration points is huge. Furthermore, most of the central Swedish systems were never developed to be used in an international context, something that has incurred increased costs. It is estimated that about two thirds of the expenses for the multi year project can be contributed directly to internationalization of central Swedish systems.

The top priority of the project has been a fast completion. The customers’ have been eager to start using the cash-pool solution, since it both gives better control over the total liquidity and reduces the risk associated with manually transferring large amount of
money between branches. In order to swiftly respond to customers’ core needs the bank has chosen to employ a phased roll-out. In the first phase the system covers main markets and handle the most commonly used types of liquidity. Other markets and types of liquidity will follow in a second phase to complete the system.

14.5 Customer rating
The customer rating system was developed in order to meet the European Union directive discussed in an earlier chapter. The main function of the system is to calculate the capital cover rate of each customer and assign a business classification type, such as “private person”, “insurance company” or “state”. Customer data is retrieved from PARI before calculations are made and updated information is then entered back into PARI using a special update service.

Development were characterized by time pressure since the requirements, due to the law making process, were not fully known when the project started and once the law was passed the system had to be operational in very short time.

For the PARI access the technique of using a special communications cobol-module have been employed. The module downloads the party and stores it and other modules can then access the party locally. In addition to the advantage of having the access logic collocated, the access module also makes it easier for business logic modules to extract the pieces of information needed for calculations. The downside with using the module is that it is harder to isolate and trouble-shoot errors in the system. When using the module it is more difficult to determine where in the process the error occurred, since so many business logic modules potentially may introduce a fault in the data stored in the module.

In addition to updating PARI the customer rating system also provides a set of lookup services for consumers which needs to access rating data. The system also uses the Service Concept to realize its services, but administrators have taken a different approach to service specification. Essentially the idea is to let the first consumer propose how the service interface should be designed and then hope that others will appreciate the functionality and reuse the service. In order to make reuse more likely the administrator usually enhances the consumer’s suggestions with related functionality that is expected to be useful for others. In reality, however, only a minor level of reuse is accomplished.
Analysis

15 Aspects of granularity
This chapter contains the first part of the analysis, which regards service granularity. The first few sections will focus on how granularity affects IT-complexity and then an analysis of other impacts on various quality aspects of software will follow. The chapter concludes with a discussion about the trade-off that exists between reach, range and realm.

15.1 Indications of complexity
The dictionary definition of the word complexity, used earlier, referred to something as being made up of several complicated parts or made up of parts where the relationship between the parts is imperfectly known (Merriam-Webster, 2006a). In Handelsbanken’s IT environment both types of complexity are at all times present.

The individual systems in the environment are by nature highly advanced because they make up such a vital part of the bank’s capability to do business and interconnecting them will, as a consequence, always be a complex issue. Further adding to complexity is the bank’s grand history of IT-support for core business processes. Over the years, systems from different epochs of the computing history have accumulated and this has created a complexity of its own kind.

It is also easy to observe the kind of institutional complexity that is always a part of large organizations. One example is the difficulties encountered in association with the banks internationalization. Both PARI specifically and the IT organization as a whole still focus on the Swedish business; this is not because it is in any way impossible to create a more internationally coherent IT support from a technological point of view, but rather because it is difficult for the IT organization to focus on so many unique needs at once and for the business organization to coordinate orders for IT-support between highly decentralized units.

What the complexity issues boil down to is the enormous amount of information that the bank needs to process and record in order to complete its commitment to the customer. A name may be enough to identify the customer at the local branch office, but due to regulations and practical matters an approximate total of 300 attributes are required in PARI just to describe the identity of this single customer. If the customer opens an account, takes out a loan or pays a bill, the amount of information required grows accordingly. The number of processes which creates such information is also an expression of complexity. At the end of the loan process, the money must show up on the customer’s account so the customer can use it in the withdrawal process or the bill paying process. All steps are supported by IT-systems that generate and pass on information about the customer and the transaction. The number of systems that interconnect in this way is estimated to about 500, a number that in itself is a manifestation of complexity.
15.2 How the dimensions of R3 and complexity is related

The following section will focus on how granularity of both the architecture itself and single services within the architecture affects the total complexity. The analysis is divided into reach, range and realm.

15.2.1 Reach

*Reach* was in chapter 10.2 defined as the extent of locations and people the IT-infrastructure is capable of connecting. A number of observations where the reach has had a major impact on the service granularity, and the related complexity, have been made in the investigation. Customers have for instance direct and explicit demands and wants increasingly to enjoy “one bank, one system”, regardless of national borders or what kind of services or products in question. In order to meet customer demands, Handelsbanken has to rework a vast number of processes. The customer wants, for example, to have a complete overview of its liquidity, which may seem like an easy task to accomplish, but as shown above, the process involves many different systems and parts of the IT-infrastructure. In other words, to accomplish what customers may experience as an obvious and wanted service may in fact be very complex to achieve.

It is not just the liquidity that the customer demands a single access point for, but progressively the bank’s whole commitment to the customer. This was one of the drivers behind the PARI implementation – the customer wants the bank to only use one registered address, regardless of engagements with different branches or business units.

Customer demands drive the reach outwards in the R3-model beyond domestic borders despite that most IT-infrastructure is owned and operated by the Swedish retail banking organization, which historically has been the core of Handelsbanken. However, a boost in mergers and acquisitions over the last years has created some political hostility between Swedish and international organizations since large business projects that incorporate IT-infrastructure tend to favor or prioritize Swedish organizations, often due to business pressure. One such example is the roll out of the cash pool project. This is not surprising or foolish since most customers are still Swedish and most business is done in Sweden.

Nonetheless, the Swedish organization’s dominance implicates that the International organization cannot use the functionality of shared general infrastructure components as was intended, and therefore has to use different special solutions which only add to the overall complexity. One way to approach the problem can be seen in the case of the currency transfer system, to let the International organization own the system in order to achieve the desired reach. In this case there were apparent possibilities for economics of scale if Handelsbanken only had one currency transfer service.

Aside from explicit customer demands, one must take into account legislative restrictions, which tend to demand a greater reach. Furthermore, there are many technical barriers that preclude the freedom of choosing a desirable reach. Most of the technical restrictions tend to press reach downward in the R3-model, into smaller services.

As Handelsbanken has acquired companies and established new international branches, there has been a conscious choice not to rip and replace the IT-infrastructure already present. The result is an overwhelming variety of IT-systems, in many countries, running on different platforms. This is one of the classical reasons for ending up with a complex architecture made up of disjointed legacy systems.
So far the discussion has focused on the reach of the entire infrastructure, but *disjointed legacy systems* also imply restrictions on individual services. PARI does not, for example, support international applications working online, due to technical problems on crossing platform borders. The reach of PARI also excludes many systems at HCM that are not developed in-house and run on different platforms. The purchased systems represent complexity in form of *disruptive innovation* and are justified by the need to stay competitive. It is too expensive and time consuming to develop all business logic in-house and therefore front office systems are bought of the shelf and integrated into the infrastructure already in place. There are technical barriers that prevent the purchased systems from using PARI services, and fully utilizing the infrastructure at Handelsbanken; therefore a lower reach has pragmatically been chosen in the case of PARI. The conclusion is that the choice of reach on an infrastructure level has implications on the reach of specific services, and conversely the choice of reach of a service may dictate the reach of the infrastructure.

### 15.2.2 Range

The range of an architecture, system or service refers to the *functionality*, expressed in terms of business activities that can be completed and seamlessly shared across the level of reach. As processes evolve, different ranges may be needed and the difficulty is to find an appropriate level of range for the reach and realm used.

One example of such volatile processes is the managing processes for derivatives, which change continuously due to updated products. These processes require IT support because manual handling is too error prone. As a result, the supporting IT-systems need to be highly flexible to accommodate for new products or, in other words, the range of the systems will need to be continuously increased.

In order to reuse a service, the offered functionality – the range – must be appropriate for use in more than one service. This is theoretically a mere exercise of logic, but in practice it becomes a little less clear cut. One of the problems is that business managers are not always able to agree on a proper range for a service, especially if usage scenarios differ slightly. The different units and projects will always argue their uniqueness to get a service tailored to their needs. In the case study it was, for instance, observed how the insurance unit was very far from the capital markets unit in terms of customers, products and general requirements for IT support. One interviewee argued that whatever synergies could be accomplished by sharing services between these two units would be lost due to overhead in administration of the shared service. Regardless of whether that particular statement is true in the general case or not, it can still be concluded that, even though the functionality required by two processes seems to be compatible, a shared service is not always desirable since the business context may vary greatly.

PARI does not implement the entire range of functionality required by a complete party system. Due to time and cost restraints it was decided that some functionality should be left out. One example is that the system does not handle organizational units of customer companies. Few processes in the retail bank’s business require knowledge of corporate structures and the business pressure generated by the impending regulatory demands called for quick solutions. In the capital market’s business, corporate structure is essential to functionality and part of the reason why HCM still uses the old pool system instead of PARI is that this functionality is lacking. The point is that, even
though the range for a specific service can easily be reduced, the functionality per se will have to be implemented somewhere. The business need defines the range of the infrastructure and once this range is fixed, adjusting the range of services will only be an exercise of moving functionality around. Scoping out functionality, i.e. to leave some piece of infrastructure range outside the range of published services, is generally a bad idea, since the functionality will then need to be realized in an unsupported manner.

Findings from the case study also indicate that too great a range can be problematic from a developer perspective. In a highly specialized organization, programmers cannot be expected to learn the internal workings of a service or the business context it is used in. The problem increases when programmers are not only expected to grasp new concepts and routines, but do it without training or readily available documentation. In the PARI case it seems like the lack of this kind of knowledge management resulted in overly complex modules and in one instance even erroneous behavior. One way to manage the need for greater understanding of the business context is to integrate more business architects in the development process to act as the bridge between programmer and orderer.

Another way to simplify work for programmers is to make the services smarter. This means extending the range of the service to filter the result. For example PARI could be programmed to filter protected addresses and other sensitive data, so programmers will not need to consider these special cases and can concentrate on solving the problem. Adding this functionality must, of course, not restrict usage of the original functionality. It must still be possible to retrieve a protected address if the business usage requires this.

The reach of the functionality on an infrastructure level can dictate the reach of an individual service. If, as in the PARI case, all actors are expected to share one service, the entire range of functionality needed across the enterprise will have to be implemented in this single service. If, on the other hand, the need of functionality as prescribed on the infrastructure level is limited, there is no general benefit in building a service with greater range, since services then may overlap in functionality and complexity consequently will increase.

### 15.2.3 Realm

As long as complexity is under control, it seems like business benefits generally increase when either the reach or the range is increased. In the case of realm, on the other hand, it appears like both ends of the scale are beneficial. The greatest business benefits come from either having very specific services, i.e. a small realm, or very broad services, i.e. a large realm.

Having a very large realm means trying to cover as many kinds of functionality as possible in one service. It can be perceived that services with a large realm are less complex and more flexible than smaller services, but this perception is deceiving. No matter how ambitious the development is, there will always be business critical realm which is not possible to incorporate in new systems or services. Another problem of including everything in the same service is that processes made up of such coarse-grained services tend to become static. The flexibility and adaptability of the process is dependent on that the services can be easily changed, replaced and re-orchestrated.
The antithesis is to break down the realm into very small and fine grained parts. This was done by Handelsbanken in the past and has resulted in an IT-architecture with over 500 systems.

A problem with the small-grain approach is that the parts often become overlapping and hard to maintain and manage over time, resulting in increased complexity. Another closely related problem is that a service cannot be too small, since it would generate an overwhelming amount of dependencies in the IT-architecture. If services are too finely-grained they may also be impossible to deploy in any context or process that they were not originally designed for and no reuse will be possible. The result will be a hard wired IT-architecture composed of many parts, but still static. Hence, neither extremely high nor low service granularity is correct but rather somewhere in between in order to build a loosely coupled architecture that will be flexible and support the ever changing business environment.

Furthermore, if a service is too coarse-grained it becomes difficult to utilize and comprehend, as numerous programmers have expressed. One possible approach to deal with oversized services is to divide the realm in the manner explained in section 14.1.8. In the section it is concluded that different services for customers and parties of Handelsbanken would probably be a good idea. A further partition of customers is, however, not recommended, since it would require the service consumer to be familiar with too much context and thus make the service less reusable and the overall architecture more complex and also more static.

Another example in which the context of the service constitutes a major predicament was when team members in a project at HCM found out that an Instrument is a Financial Markets Offer according to HBDM. However, there may be many instances of FMO which all defines different entities depending on the context. In response to this PARI has three different manuals covering the same service but for service consumers of different contexts. The main reason for this is to keep the actual realm of the service as coarse-grained as possible but make it comprehensible and understandable to various professionals using the service. Thus, many of the business analysts’ and IT-architects’ problems were solved or at least eased, but the programmers still had to use the coarse-grained service and complained about information overload and as argued above, it is probable caused by improper knowledge management rather than unskilled programmers.

In order to not create a too complex service in the case of the Cash Pool-project a choice was made to limit the realm and for example not include all kinds of liquidity. This is however a slightly unusual course of action since the business demand of a certain realm will not vanish if it is not implemented as a service (or part of), the choice is more often about how many services are desired to cover the business demand, and in fact, as the Cash Pool project continuous more realm is planned to be implemented.

15.3 How granularity affects other quality aspects of Enterprise Architecture

Service granularity affects many different characteristics in an Enterprise Architecture as argued before, not just complexity. It is therefore important to design services not just out of the aspect of reducing complexity, but rather consider the whole impact of a certain service granularity. There are also many occasions where service granularity
cannot be isolated to a single axis but affects the complexity as whole or is indifferent to which one of $R^3$ is altered, the important point is rather that the volume, the service granularity, is decreased or increased.

15.3.1 Accuracy
PARI’s mission is to manage party information of all kinds of parties and for almost everyone and every system in the corporation. Thus, the system is a part of a vast number of business processes, and intentionally so, however, this has caused unexpected problems. One such problem is the inaccuracy of the stored party information, which inexorably decreases the usage of the services.

With increasing $R^3$, the difficulty to keep control of the stored information consequently also increases. Small service granularity, a fine grained service, for example a customer service which just serves and holds information for one single branch office, would be much easier to comprehend for all kind of involved professionals, compared to a customer service serving the entire corporation. Users of the smaller service would not have to consider different formats for addresses, in contrast to the coarse-grained service which must handle all kinds of international address formats. It is obvious that local routines and guidelines may arise when the users do not have a clear understanding of the entire context, which is reasonable. How and why should the local branch office know how an insurance payment system prints addresses?

Another aspect of the example above is the ownership and responsibility of the information in different services, a classical CRM domain. If services are divided in $R^3$ what happens when the same information is needed in several services simultaneously or when information needs to transcend service borders? The result is many duplicated local copies of the information, which is very problematic since it is hard to keep all information accurate, coherent and up to date. A problem usually referred to as a problem of Master Data. The solution at Handelsbanken was, as the reader should be familiar with, to consolidate all party information in one single coarse-grained service. But, as will be further emphasized below, PARI has not taken the full responsibility to guarantee that the information stored is accurate.

There are many times hard for the developers to penetrate a system design in order to discover errors, since they do not have the insight into the information stored in PARI or the business-context thereof. Hence, the programmers would need more comprehensive test cases, since they cannot be expected to be knowledgeable of all aspects of the business context. However, that is something that can be expected of PARI and therefore should be supplied by PARI in order to keep the party information stored accurate.

15.3.2 Understandability
The sheer amount of information in PARI makes it almost ungraspable, which could be handled by decreasing the scope of PARI in any of the three dimensions of reach, range and realm. Before all party information was consolidated in one service this was likely a lesser problem, since the information was stored and used were it had a context and mattered. In case of PARI on the other hand, the control or understanding of the usage of the information is consequently limited and therefore focused on supplying party information. One approach as described earlier is to divide the service in more services and thus decrease the service granularity. However, that might result in a reversed problem of the same kind as information overflow. Many small services creates a service
A governance problem, it could be hard to grasp the amount of services, understand and locate what each and everyone does, and problematic to deploy the appropriate or “correct” service.

In contrast, coarser grained services increases the need for appropriate documentation since it becomes harder to comprehend both the information stored and the service itself. Larger range requires more detailed explanation of what the service actually does and how to use it, larger realm demands a better understanding of the entities in question and finally larger reach entails the documentation to be written for the appropriate target group of readers. Furthermore, as noted in the investigation service consumers has to contact the service providers in order to attain the documentation, an approach that works for fine grained services but is non-scalable and not recommended for coarse-grained services. The researchers have found that a usual approach to disseminate knowledge in Handelsbanken is by meetings, also an improper approach for coarse-grained services.

15.3.3 Reusability

In order to create reusable services the dependencies on the environment should be minimized. If a service is indifferent to what country a customer is resident in, that service will be more reusable than a service which requires Swedish customers to live in Sweden or a service that works with “signing party”, rather than for instance “customer signature” will be more reusable. In other words, coarse-grained services are less dependent on their environment, since they are less specific, and therefore more reusable. On the other hand, coarse-grained services may be too large in terms of R^3 and become too complicated to use or figure out which will decrease the actual usage of a service.

To be able to create reusable services require insights in future business demands, which of course is not an easy challenge. The higher abstraction level, more coarse-grained, the harder it gets to foresee the upcoming demands, since more service consumers have diverse desires in all dimensions of R^3. Consequently, smaller services of lower abstraction level are more well-defined and it is therefore easier to capture tomorrow’s demands.

15.4 Consider all three dimensions simultaneously

It should by now be clear to the reader that all undertakings of IT must have a business purpose; IT-maxims derive from business maxims which in turn derive from the organization’s business strategy which ultimately forms out of customer demands. The IT-maxims sets the goals and rules for IT-infrastructure and the organization’s view on IT-infrastructure influence how services are implemented. However, the R^3-model applied on the IT-infrastructure does not dictate a certain level of reach, range and realm (R^3) of a specific service. On the other hand, the strategic aim in terms of R^3 will restrict the freedom of choice of R^3 of an individual service. As seen in the case of PARI, legislation forced Handelsbanken to include all customers in the party service, regardless of what kind of engagement the customer had with Handelsbanken. The law only stated a minimum of reach, range and realm but the corporation chose to implement a higher degree of functionality, for example covering all kinds of attributes a person could have, instead of just implement the required rating attributes.
As discussed earlier, the vision of an ultimate IT-infrastructure tends to push the R³ outwards, which implicates services of relatively high R³, or a lot of medium or low level services. There are however practical limitations such as time, money and capability restraints. The most significant rationale to decrease reach, range and realm of a service, observed in this thesis are; difficulties in development, run-time properties, and difficulties in service usage.

Difficulties in development have been noticed as politics and business pressure. As discussed earlier business managers need to agree upon the service definitions, the extent of reach, range and realm, and the more they need to concur the harder it gets. There is also a constant business pressure not to realize more than needed in the specific situation, as well as the difficulties to envision future needs that would make the service more reusable.

There are also run-time properties limiting the service granularity. More range will for example require more complicated database questions, which will require more microprocessor power, and thus cost more. It is often, in general and in the case of PARI, desirable to increase the reach, range and realm of services in order to achieve economics of scale. It is however, often irrelevant in which dimension the change takes place, the concept is rather to share resources through higher abstraction level of services and that fixed cost is relatively high compare to marginal cost of consolidating services. The main focus of economics of scale is to reduce the number of dependencies, since they minimize reuse and adds to the complexity of the architecture, as well as from a maintainability perspective as from a development perspective. Nevertheless, as noted in the investigation, only small earnings have been made since many systems still use the dispersed older systems.

It is not just service development that experience difficulties as the R³ increases, but also service consumers, including business managers, architects and programmers. The huge amount of information in PARI has for example revealed itself many times as unmanageable complexity and incomprehensible to various professionals, due to improper knowledge management. The conclusion is that the service granularity of PARI is too coarse-grained in relation to the knowledge level in the organization. This could be resolved either by working with the knowledge level or by reducing PARI in any of the three dimensions. The latter could, for example, be accomplished by dividing PARI in two services; one handling customers and one for business units of Handelsbanken, which would have decreased the amount of information to be comprehended. It would also have had implications for the services in the other two dimensions. The range of each service could for instance have been reduced since the business unit service would not have needed functionality for identifying if the business unit had a confidential address. Thus, all three dimensions must be considered simultaneously as they all affect the service granularity.

The decision of service granularity in terms of reach, range and realm should be governed by the organization’s view on IT-infrastructure. An enabling view would imply IT-maxims to stipulate larger reach, range and realm than a dependent or utility view. It is therefore important to assess all three dimensions simultaneously so that the resulting service granularity is in consistence with the reach, range and realm of the IT-infrastructure and hence in consistence with overall business strategy.

Problems arise when services are not in alignment with the organization’s view of IT-infrastructure. Business managers are not willing to pay extra buck for reach not asked
for, nor are programmers impressed by overwhelming range when trying to complete a simple task as retrieving a customer address. It is therefore vital to an organization to communicate the view on IT-infrastructure throughout the organization and then live by it.

15.5 Other aspects of SOA affecting complexity

There are of course other aspects of SOA or the Service Concept at Handelsbanken, than the service granularity affecting the complexity in the Enterprise Architecture. These characteristics address different forms of unnecessary complexity but are unaffected of the level of R³.

15.5.1 Handelsbanken’s Service Concept resembles SOA

The Service Concept, as argued earlier, does resemble SOA to a great extent, especially on a conceptual level, there are nevertheless technical differences. The Service Concept, is message based, a cornerstone of SOA, and encapsulates the inner workings of the services as much as possible in order to create a channel independent infrastructure. In concurrence with SOA it also defines a standard in Handelsbanken for information exchange between systems and services. However, the standard is not an inter-enterprise widely adopted standard, advocated by SOA. Another characteristic of SOA is that services should be stateless, which for instance PARI is, but due to technical difficulties the Service Concept is not.

15.5.2 The Service Concept is message based

A dynamic messaged based structure is the basis in making an architecture loosely coupled. The advantage, in contrast to a static structure, is that services may at any time be extended with new types of messages, since service consumers only pay attention to those messages they are already familiar with. It also means that service consumers can be replaced or developed without the involvement of the service provider. An important aspect of the architecture in order to maintain a future proof loosely coupled architecture, which does not add complexity as other services evolves.

The concept of message based implicates that the entities should be meaningful as a whole, a message should for instance be an entire order and not several messages, one for each order entry as shown in section 9.2.4. Coarse-grained services tend to be more reusable due to minimized dependencies on the environment, as argued above. Messages are one such constraint on services. The services consumer knows the context which the information is to be used in, contrary to the service provider. Consequently, in order to produce reusable services the elimination of unnecessary information should be done on the consumer side.

15.5.3 Internal versus external standards

Standards are more or less a prerequisite in order to achieve interconnectivity and interoperability in an IT-architecture of many systems. However, that does not imply that the standards needs to be global widely accepted standards. As long as everything within the boundaries of the architecture complies with the same rules and guidelines, it should be no problem of achieving a coherent architecture and there is no need to obey external standards. Nevertheless, problems may arise when the boundaries of the architecture are expanded. Mergers and acquisitions, business expansion and innovation
continuous to extend the architecture as well as internal business and technical development do. It could prove problematic to incorporate new features if the internal standards differ from those used outside the architecture. It is therefore vital to a company to assess how different the internal standards may be from those external, IMS is for example widely used in the financial sector and not unique in Handelsbanken.

The Service Concept used in Handelsbanken is not implemented with Web Services, since it was not an available option at the time. Web Services has however gained acceptance and today become the standard most vendors promotes. The shift from developing almost every system in-house to a more diversified approach, with packaged software from specialized vendors has challenged the boundaries of the architecture and it may be time for a re-evaluation of the chosen standards in order to ease the effort of integrating new software and thus reduce, or at least not increase unnecessary complexity.

15.5.4 Common message data model adherent

One of the reasons for developing PARI was the ambition to supply the corporation with party information from one single source, which would be easy to maintain and update, making the information more accurate and consistent. Furthermore, services have to present correct and sought for information to the services consumer, in order to make the service successful and reusable. In the case of PARI, as seen in the investigation, the accuracy of the information is not enforced by PARI but delegated to systems updating PARI and underlying data sources, creating a problem of Master Data. The “single source” approach does tackle the problem of dispersed local copies but does not address the accuracy of the actual data.

SOA stipulates that services should adhere to a common message data model, as illustrated in section 9.2.4, to guarantee that each and every service interprets the information in the same way, make them reusable and decrease the complexity in the architecture. In concurrence with SOA, Handelsbanken has implemented HBDM and HBOM to fulfill this goal. However, the model works fine for classifying information, but lacks rules for formatting. Resulting in phenomena like services that do know how to communicate, know the difference of price including or excluding VAT, but does not know if the currency name should be spelled out or abbreviated. The problem is not, as seen in the address example in the investigation, the construction of PARI or HBDM rather the usage thereof.

Nonetheless, semantics is often presented as one of toughest task in realizing an SOA and a major contributor to complexity. Handelsbanken has managed to define the entities in the architecture and the meaning of them in a context and thus reduced complexity considerable. To further reduce complexity data should be reviewed and conformed to an enterprise wide set of guidelines on entry as well as on presentation, which is in alignment with the company’s choice of a reach for all systems in the corporation.

15.5.5 Encapsulated

A major characteristic of SOA, PARI and the Service Concept is to encapsulate underlying logic in order to increase reuse and reduce complexity through looser coupling. One could naivety think that agreeing upon one single data model such as HBDM would be the best way to reduce complexity and promote reuse of services, a very
tight coupled architecture. Development in such an architecture might be easier and services might be extensively reused. However, the framework would be so strict and tight that it would be impossible to integrate any new or old systems without completely rewrite the entire code, as seen in one example in the investigation.

In the light of the strategy to align Handelsbanken with accepted standards and an increase in packaged software, reuse and managing complexity has to be achieved through an encapsulated loosely coupled architecture, which PARI aims at. Nevertheless, the selection keys used to address PARI, does not entirely encapsulate the underlying data resources. The second key, describes how PARI should carry out the request. Developers can choose to just look in Swedish databases, which imply that programmers do know the underlying structure of PARI. Since PARI stores different types of information for parties from different countries, the programmer has to know what kinds of attributes are available on a specific party exposing the inner workings of PARI.

To reveal the underlying logic does not just tightened the coupling in the architecture and making complexity management more difficult but also adds complexity for service consumers, which has to understand how PARI works in order to use it efficiently.

### 15.5.6 Stateless and invocation style neutral

SOA strongly recommends services to be stateless and invocation style neutral to minimize dependencies on the environment and to imply as little features of the service consumers as possible to facilitate maximum reuse. PARI is stateless, but the Service Concept is not, which as noticed in the enquiry, excludes systems at Handelsbanken Capital Markets to use PARI.

Another aspect of environmental dependencies is if the service is designed to be invoked in a specific manner such as batch processing, request/response or asynchronous queuing. PARI is designed for online usage of relatively small questions in a request/response manner. However, PARI does not excludes the possibility of batch processing, but is rather inefficient for large batches and would need special services for end-of-year book closings. Invocation style has a further dimension in the financial sector, namely channel independence. A service designed particularly for dialogue usage by a bank teller is of little use when developing a web portal for customers, hence reuse would be limited and complexity would increase since more services would have to be implemented and more relations realized. PARI is however completely channel-independent.

### 15.5.7 Aspects of complexity not fixed by SOA

It is important to remember that SOA will not, however promising, solve all problems in a corporation. SOA will on its own neither create any new competitive advantage nor business value, nor will it manage or reduce all complexity of the IT-architecture. Some complexity, as argued above, is necessary and adds to the company’s capabilities and ability to distinguish itself from its competitors, hence should not be reduced. Necessary complexity arises from intelligent business operations and will of course not be decreased by SOA.

An example from the insurance business will further augment the statement. Insurances and life pensions are bottom line a valuation of risk, the likelihood that something
happens and the probable cost thereof, such as when you will die or your property be stolen. The monthly fee is calculated by complex mathematical formulas, based on a variety of information, which is something IT-systems must do in order to supply millions of customers. Hence, IT becomes very complicated, but it is sought for complexity that in the end generates business value.

Finally, sometimes it easy to be carried away and lured by SOA advocates and vendors into thinking that SOA will solve everything, be the one and only business solution or even make your coffee in the morning. This is of course highly questionable, organizational routines and politics are for instance unwise to solely assess by an SOA initiative.

16 Handelsbanken’s IT view

The presented theory base points to the insight that a corporation’s strategic view on IT is related to how extensive services can, or should be. In this chapter Handelsbanken’s view on IT and how systems are affected will be further analyzed.

16.1 Handelsbanken is dependent on IT

Weil and Broadbent (1998) presented a spectrum consisting of four “views” a company can take on information technology; none, utility, dependent or enabling. The general view has a broad range of implications, especially relating to the extent of services. It is therefore helpful to establish which view Handelsbanken uses.

Handelsbanken has to a large extent taken a centralistic approach to IT. The central IT department maintains most IT related functionality for Swedish business units and also has a relatively large impact on international units. Therefore, clearly, Handelsbanken does not have a none view on IT. On the opposite side of the spectrum is the enabling view. The vice president outright dismissed this view and argued that overspending on IT to build a potentially useful capability was not a strategic option in Handelsbanken.

Left to analyze is the utility and dependent view. One could argue that because of the focus on cost efficiency Handelsbanken has a utility view on IT. However, given the relative cost mass that IT represents in conjunction with department head’s representation on the board of directors and the fact that the outcome of strategic business ventures to a large extent depends on the IT capabilities at hand, it seems like a dependent view is a more accurate description of the state of affairs regarding IT in Handelsbanken.

One should yet notice how the impact of the relatively recent change in strategic direction regarding IT-infrastructure questions, have positioned Handelsbanken’s IT view further away from the utility view than earlier. In the previous organization the IT department was not mandated to own infrastructure and only had the role of supplier. The business was responsible for financing infrastructure directly and consequently most parts of the infrastructure were realized through business application projects. Since the business in Handelsbanken is much decentralized, so were decisions about infrastructure and in practice the development of infrastructure were governed by individual business unit needs. The new direction, as explained by the vice president, indicates a shift towards more corporate level owned systems and services and with that also a greater effort to create a more flexible infrastructure.
16.2 Deals based decision management

The focus on business project driven development of IT has generated a culture of deals based management. Even though there is an architecture function with responsibility of overseeing the development of the infrastructure, it has often been problematic to raise money to work proactively with these issues. Investments in infrastructure have, hence, not been governed directly by strategy, but taken the route via business unit managers and their perceived needs.

Management by deals is in general not consistent with a dependent or enabling view on information technology, which requires a more centralistic view on IT investments. Given the recent change in strategy, it is likely that more infrastructure investments will be justified directly by strategic concerns in the future.

16.3 How PARI was affected by Handelsbanken’s IT view

The development of PARI was driven by the customer rating initiative. Regulations had demanded of the bank to accurately and consistently rate all its customers, regardless of weather or not the customer had engagements with the bank in several separate countries. To comply with the regulation the bank implemented two new systems. The first one was a single unified system, PARI, responsible for aggregating customer information from all available information sources and make sure that each individual customer appeared exactly once in the aggregate. The aggregated information was then used by the second system, the customer rating system, to identify the customers’ engagements in different countries, which was a prerequisite for accurate rating.

The rationale for separating the two functions was of course that the aggregate of the customer could be reused in any place where customer data was needed. Making it possible to retrieve all needed information from one single source would greatly reduce complexity in any system which, for instance, managed international customers.

The regulatory demand, in combination with the bank’s international expansion strategy, put constraints on the required functionality of the bank’s IT-support. Functionality with a minimum realm of “customer”, a minimum range of “creating a rating” and a stipulated reach of “all countries where the bank is active” would have to be realized somewhere to comply with the regulations. Handelsbanken chose to split the functionality, as described above, into one infrastructure service (PARI) and one business service (customer rating system). For the service PARI the dimension of reach was thus stipulated by the reach dimension of the IT-support.

The realm of the service needed only to be “customer” but was chosen as “involved party”, with the intention of making the service even more reusable and further reduce complexity. The range was allowed to vary, within the service, depending on reach and realm. Consequently there is one set of data available about all customers worldwide, another set of data for the subgroup represented by Swedish customers and a third set of data for the realm of Handelsbanken units.

The dataset for Swedish customers is larger than that for worldwide customers, because PARI is intended to be the primary data source for systems managed by Swedish business units. Using a single service as the sole source of data puts implicit demands on the range of that service to be equal to the total range of functionality in the IT-support, as requested by all Swedish business units. However, since the party data is so extensive
the entire range has not yet been implemented. One notable example of missing functionality is corporate structures.

The failure to provide the entire range of functionality creates complexity, since systems that depend on the missing functionality will have to access one or more of PARI's data sources directly. Further and more unexpected complexity is added by the large R³ volume of PARI. The service is among other things described as difficult to understand, hard to use and more expensive than the older alternatives. The analysis, however, show that scaling down PARI is no obvious solution since many other values would then be lost. The source of the problem is not PARI in itself, but the unalignment between PARI's abstraction level and the available resources in terms of time, money and knowledge.

At the time PARI was built the bank’s view on IT was dependent, but leaning towards utility. There were no separate infrastructure initiatives and PARI was built in on a tight schedule, economically justified by the customer rating project. The deals based decision making process gave stronger business units, most notably the Swedish retail banking units, a strong influence over priorities in the project and as a consequence the realization of the complete functionality was postponed once the stronger units had gotten their share.

The view of IT in combination with the deals based justification of investments can also be seen as responsible for the lack of knowledge resources. PARI was at its introduction not just-another-system, but the first of a new kind of systems. The large granularity increase responsibility for understanding the business, but tightly timed business projects provide little opportunity for studies. Given the programmers accounts of their first encounter with PARI it is apparent that an education initiative has been missing; recall that one programmer had not even been distributed a manual.

If PARI had been justified on its own merits by strategic concerns and created as part of infrastructure in a proactive manner many of the observed problems could probably have been avoided. Economic justification by a strategic intent of “providing the customer with an experience of a single coherent system” would for instance likely had offset the power balance between financially stronger and weaker units, resulting in a more complete functionality in terms of range and consequently a lesser complexity of the overall architecture. The strategic goal for infrastructure, as stated by Mr Johansson, “to create the effective IT delivery” would likely justify investments in knowledge and education, since leaving personnel to figure out new technologies on their own as part of business projects hardly can be seen as “effective”.

17 Conclusion

This chapter summarizes the results of the thesis and fulfills the objective as stated in chapter two. The first section contains a short recapitulation of key findings from the analysis and the second section shares the researchers’ conclusion.

17.1 Synthesis

The scope of an SOA service – its granularity – can be precisely described using three factors: reach, range and realm where reach defines the locations and people the service is capable of connecting, range defines how much functionality the service offers, and
realm defines *what kind of functionality* the service offers. The granularity is defined by the volume demarcated by reach, range and realm. In general, business benefits from SOA increase with increased volume, as long as the organization manages the resulting complexity. Key drivers of business benefits, relating to enlarged service scope, are an increased reuse potential and a decrease in the number of relations between services.

Organizations can take a view on IT ranging on a scale of none, utility, dependent and enabling. The view reflects the strategic importance of IT in the organization, both in terms of the opportunities IT is expected to create and the commitment to IT the business organization is willing to make. There is no more-is-better relationship between IT view and business benefits; instead the appropriate view is defined by the role of IT in the organization’s business strategy.

Complexity is the state of a whole made up of complicated parts or parts where the relationships are imperfectly known. Managed complexity generally adds to business value, as the complex entity is difficult for competitors to imitate, whereas unmanaged complexity destroys business value. Service-oriented architecture becomes more complex when the volumes of incorporated services are either too large or too small. Services that are too large tend to be complicated to build and use, whereas overly small services tend to generate an unmanageable amount of relations.

Exactly what is “too big” and “too small” is context-dependent, as complexity arises from a number of environmental sources. In the case studied, *business pressure, politics* and *neglected knowledge management* have all been found to be sources of complexity relating to service granularity. The organization’s ability and willingness to manage complexity arising from these sources depends on its commitment to information technology – the organization’s view on IT.

In organizations with a *dependent* or *enabling* view on IT, investments in information technology is generally driven directly by business strategies using business and IT maxims. In these organizations, proactive investments to reduce complexity, such as building infrastructure services or training developers, can be undertaken and justified by strategy alone. In organizations that have a *none* or *utility* view on IT, and which use a deals-based justification process, the opportunity to work proactively is smaller. Consequently, organizations with a depending or enabling view on IT are able to manage greater complexity, reaping the benefits of larger grained services.

### 17.1.1 Bottom line

The objective of this thesis has been to factorize the notion of service granularity, explicate how the factors are interrelated and explain how an organization’s strategic view on IT affects the optimal service abstraction level in a Service-Oriented Architecture. Hence, the bottom line of the thesis:

The organization’s strategic view on IT affects the amount of complexity the organization is able to handle, limiting the optimal SOA granularity as described by reach, range and realm.
Discussion

18 The larger picture

Given the relatively strong impact of business strategy on a single service’s granularity, and with the distinct features of Handelsbanken’s IT organization in mind, it is inspiring to take a step back and attempt to grasp the whole picture. The study does not try to cover any terrain beyond what is specified by its objective; still, the authors have spent a half year studying SOA, IT-strategy, and the practices of Handelsbanken, and have formed an understanding of the larger picture. This section conveys conclusions from observations on the very border of the topic, and places the question of service granularity in a larger context.

We began with the assumption, as stated by Moore and others, that companies must untiringly innovate and evolve to stay in business. In most cases, corporate evolution encompasses adjustments, or radical reengineering, of business processes, and because virtually all processes in a modern company are supported by IT, the conclusion can only be that information technology needs to evolve at least as quickly as the business does.

At this point, however, the core/context problem makes itself clear. Moore divided innovations into core innovations, i.e. new innovations that provide competitive advantage, and context innovation, i.e. yesterday’s innovations that now represent a ‘must do’ for every company in the business. The problem with IT development, especially in the financial sector, is that it creates a huge amount of context innovations in the form of legacy systems, which are very expensive to replace.

Imagine, for example, the systems at Handelsbanken’s insurance unit. Wired into the insurance systems are rules governing the policies of hundreds of thousands of life insurance customers. Once the life insurance policyholder has made the first installment, the insurer is responsible for maintaining the plan until the policyholder is either deceased or retired. Given the amount of policies and the complexity of individual policies, it is obvious that a manual transfer of policies from one system to another incurs a great business risk. The systems must therefore be maintained until the last plan has terminated, which is literally a lifetime. In other businesses it might be an option to outsource the maintenance of products invented and sold half a century ago to some third party specialist, but in the financial sector these products still make up a vital part of the business. A checking account does not, for example, provide any kind of competitive advantage, but is nevertheless a key ingredient in forming the banks value proposition. Customers put their salary into the account and use the money to consume other bank services, like credit card payments and house loans. The consequence is that essentially every IT-system needs an integration link to the subsidiary ledger which handles the checking accounts. Once integration starts, the last hope of ever replacing the system usually dies, since replacing the system now also entails the risk and the cost associated with rewriting possibly hundreds of other systems. Complexity inevitably welds old systems to the architecture.

Every new system that is introduced needs to be connected to the legacy in order to function, which makes IT development both slow and cumbersome. For instance, in the cash pool project we observed how a majority of resources was spent on updating legacy application to work in an international environment. The problem can, as argued by
Weill and Broadbent, be eased by investing in IT-infrastructure, such as a common technology base, increased knowledge management or standardized infrastructure services. Proactive infrastructure investments have the potential to take care of many issues before they start to restrain business project. The problem is deciding in what infrastructure to invest; non-utilized infrastructure will only add to legacy and provide no tangible benefits.

The best option for companies that are dependent on IT for their business is to directly justify any infrastructure investments by their contribution as the realization of the current business strategies. Since strategic planning is a long range planning model it, should be possible to have a high degree of foresightedness in the investments. Unfortunately, business strategies are, at best, intentions of action and according to Mintzberg's line of argument and are seldom completed as intended. Therefore, a great deal of flexibility needs to be built into IT, as well as into the business organization.

Handelsbanken’s core strategic principle is that the bank has “one goal and two means”. The goal is to accomplish higher profitability than the competitors’ average and the means are to have lower costs and more satisfied customers. The customer satisfaction is created by the decentralized organization, with a high level of intimacy between the local branch office and the individual customer. The strategy has worked very well – Handelsbanken has beaten the competition for 34 years straight.

For the major part of the bank’s history, however, the customer’s behavior was very static, at least from an IT perspective. The customer visited the branch office, used mail, or spoke to a teller on the phone to complete transactions. The teller, or a back-office representative, was responsible for transferring the customer’s request to the computer system; for the first three decades of computerized automation, IT never met the customer directly, with the exception of automatic teller machines and a few corporate services. The primary focus of the IT-department consequently became to keep costs down, and because the business units acted as mediator between the end customer and the IT department, all investments were justified by the BU’s needs. Given the relatively static customer demands, the practice presented few problems and, more importantly, it was a logical consequence of a business strategy based on cost savings and customer intimacy.

In the last decade, however, a dramatic change in customer preferences has occurred. The customer is no longer static but, from an IT perspective, dynamic and increasingly demanding. In the late 90’s, the Internet, along with various other channels such as automatic phone banking and mobile banking, became hugely successful. For the first time in history IT truly met the customer face to face, and did so with dramatic consequences. Furthermore, for last 20 years globalization has increased dramatically. The bank has joined many of its customers and expanded abroad, noticeably increasing the need for new IT integration solutions.

During the time that Handelsbanken has operated, customers have learned that the bank is both a reliable and flexible partner; the local branch is always on its toes, assisting the customer in managing her business. The challenge is that the customers expect the bank to be equally helpful and flexible regardless of channel. Simply put, the customer expects the bank’s IT to provide the same degree of service as the branch office, only faster and more accessibly. The strategic overweight toward cost savings must thus be reduced, and IT must become more intimate with the customer. IT will naturally never compete with personal contact when it comes to complicated transactions, but
must still, in an increasing pace, support new demands for on-line everyday transactions. If new features are not introduced quickly enough, customers will grow impatient and perceive the banks service offer as mediocre compared to speedier competitors; this thinking especially exists among corporate customers, where introductions of new functionality in the bank may substantially improve bottom-line results for the customer. The priority on time-to-market, as mentioned by the bank’s IT-strategy, is a logical reflection of the bank’s core strategic values. The consequence is that IT must become more flexible and less complex.

We have already established our view that a solid IT-infrastructure is a prerequisite for building competitive business functionality and that service-oriented architecture is one of the most promising ways to create flexible IT. The question about service granularity, however, still remains. The study provides a means for describing and analyzing services in an architecture and explains how the strategic view on IT can affect the organization’s ability to handle services of certain granularity. It does not, however, prescribe certain granularity for certain situations.

Given what has been said about business process reengineering and the core ideas of SOA, i.e. to mimic the business process flow, it seems like fine-grained services would be optimal. Small services that precisely fit the steps of the process and that can be restructured at will are ideal if one wants to swiftly rearrange a process. The disadvantage of using such small services is that reuse is very hard to accomplish and governance is even harder. If every process has a unique set of services, the process will be as costly to build, maintain and integrate had it been realized using older ideas and technology.

One natural counter-reaction to services that are too small would be to build all services coarse-grained, in the spirit of “if we don’t need it now it will sure be good for later”. The problem with this approach is that coarse-grained services, as seen in the PARI-case, have a tendency to useable by many, but perfect for none. Furthermore, a coarse-grained service will have a lot of dependants and is therefore hard to replace or upgrade. It is apparent that a middle-way strategy is needed. Our suggestion is to employ three different variations of granularity in any organization.

Core processes are highly volatile and unique. Therefore they will usually need tailored IT support that is flexible enough to change with the process as it matures. For this kind of use, finely-grained services are appropriate. Context processes, on the other hand, should, in line with the argument of Davenport and Hammer, be standardized within the organization and in some cases even between organizations. These processes are not likely to change to any larger degree and can rely on a more standardized IT-support, built on coarser-grained services that encapsulate more functionality and are shared between many consumers. Supporting all processes is a set of basic business-context independent services: the infrastructure services. These services do not provide any specific business value themselves, but supply vital functionality to other services. These services should generally be coarse grained because the functionality they provide is useful in many places and can, in some instances, not be allowed to be duplicated.

PARI can by all means be defined as a rather coarse grained service. It does not directly provide business value, but is needed by many systems to function correctly. It also supports the bank’s international engagement by collecting party data from all countries – typical properties of an infrastructure service. Piecing it all together, it seems like
PARI has the correct level of abstraction, but the bank’s view on IT has been lagging and does not fully accommodate the new customer-facing role of IT.

Furthermore, the observant reader will notice that the alternatives for dividing PARI into smaller services suggested in the analysis were not primarily aimed to reduce granularity but to make the service more evenly shaped. Today PARI provides different ranges, depending on which reach and realm one is looking at. The set of customer data that exists internationally is, for example, smaller than the set of customer data for Swedish customers only, something that has been shown to generate unnecessary complexity.

We will conclude our discussion with two questions for further research. The dimensions of reach, range and realm can be used to describe the granularity of a service and to draw conclusions about how complexity increases when changes are made to the scope of the service. Exactly how reach, range and realm should be selected for a specific service instance is governed by numerous factors. The aim of this thesis has not been to provide a tool for analysis of an architecture, but for analysis of an individual service. The results regard the relationship between granularity of individual services and the general view of IT, but cannot be used to establish exactly which services that should be implemented, how many layers that should be used and how granular services of each layer should be. A good question for further research would therefore be to analyze which underlying factors that influence the choice of $R^3$ in different parts of the architecture.

The $R^3$ framework has been derived to analyze services in a service-oriented architecture, and its relationship to business strategy has been explained using IT maxims and deals. The three dimensions as such, however, pose as quite generic. Further research may establish whether the $R^3$ model can be used as a general framework for describing services within an organization, and how its relationship to business strategies is formed in such general cases.

19 In retrospective…

The inquiry approach and the working process were laid out in the methodology part to give the reader the ability to judge the trustworthiness of the thesis for herself. This chapter contains the authors’ reflections on the reliability, accuracy and system dependability, and summarizes what measures have been taken in order to make the results as reusable as possible.

19.1 The inquiry approach vouch for good validity

Validity is a measurement of how accurate the method of data gathering is – if it really measures what the researchers intend to measure (Lekvall & Wahlbin, 2001). It is neither quantifiable nor discrete, since there is no objective answer. If a method exists that could present the true result, why not use the better method (Lekvall & Wahlbin, 2001)?

Semi-structured interviews, which give the researchers the ability to alter and rephrase questions during the interview, were used to establish that the interviewee interpreted the question as intended. To ensure that the interviewee answered on the appropriate topic the researchers always asked the interviewee to exemplify the answers. The aim
was not, in most situations, to capture the interviewee’s thought on the topic but rather what they actually did when faced with the problem at hand.

Since the thesis spans over the whole range from corporate strategy to implementation of software, a variety of professionals were interviewed. In order to achieve satisfactory validity at least two people with relevant knowledge were interviewed on each topic, excepting the vice president and head of IT department. The questionnaire was therefore, in that case, beforehand reviewed and revised by three insightfulness people who assured that the questionnaire had good face validity.

19.2 The study seems to have good reliability

Another aspect of trustworthiness of the study concerns the ability of the method to resist influence of coincidences during the interview (Lekvall & Wahlbin, 2001). As previously noted, the researchers met with various professionals, hence all presented facts are first-hand information. In order to compile an interview guide containing appropriate questions, with respect to the interviewee’s professional occupation, only some questions were chosen from the battery of questions presented in the appendix.

IT-infrastructure, systems, services and operations may be defined very differently by different people. Moreover, many concepts used in the thesis are quite abstract, such as service granularity and understandability, hence the ability to clarify definitions during the semi-structure interview was proved useful.

To make well-founded and reliable conclusions, the analysis is motivated and underpinned by established theories. The synthesis presented in the literature review has been empirically verified and seems to be true in the case of Handelsbanken; the R³-model does factorize the notion of service granularity, and observations and conclusions of their interrelationship have been made. It is the authors’ conviction that these conclusions are accurate because they so heavily substantiate acknowledged and accredited beliefs.

19.3 Comments on the ability to generalize the study

From the assumptions behind the systems approach it follows that the real systems are not usually completely comparable with each other. After the rigorous yet limited description of Handelsbanken and the elements studied, it would be contradictious to the systems approach to safely draw conclusions on any other real system. Arbnor and Bjerke (1997) point out that using the systems approach does not in any way reduce the level of ambition as compared with representative samples in the analytical approach; the difficulty to generalize is rather a consequence of the systems approach’s presumptions about how reality is constituted.

The researchers believe that the main contribution of the thesis is the development of the R³-model and the illustrative reasoning of how to use it in order to define service granularity. This theory is, as argued before, a topic that SOA or component literature has often left out, mainly because it is very industry and company-specific. The authors believe the model to be relatively stable and independent. It should be possible to use this model in almost any kind of industry or organization struggling to identify service granularity. However, the factors that actually define the granularity in each dimension, as well as factors defining how the trade-off between dimensions correlate, and the definition of the total volume are system-dependent. The key aspects of the system
studied have therefore been laid out as thoroughly as possible in order to give the reader a chance to assess the likeness of the systems for her or himself.

Nonetheless, the researchers believe that almost every large full-service retail bank with a long history shares the same key properties of IT-architecture. The most prominent inhibitor of agile development and process utilizing is complexity. Furthermore, it is important to emphasize the business context and the importance of the view on IT that the company holds. The R³-model will probably be a useful tool, but how to implement it and what conclusions can be drawn from it will vary widely.

Further, authors believe that the R³-model and the factors defining service granularity and their relationship will be of great use to Handelsbanken and hold true to other general components in the architecture. The different indications of complexity and the consequences of granularity are system-dependent, and may not be applicable outside the context of Handelsbanken. They do, however, provide an illustrative framework for what kind of factors and properties that may give a certain effect, especially in the context of the Nordic financial sector.
References

Litterature references


**Interviewees**

Andersson, N. (2006-10-31)

Andersson, N. & Wickström, I. (2006-11-22)

Börjvall, M. (2006-10-12)

Eiworth, F. (2006-12-12)

Ekberg, M-O (2006-12-07)
Elfverson, H. (2006-12-05)
Engström, K. (2006-10-26)
Flygar, H. (2006-12-05)
Fridlund, C. & Höjenberg, K. (2007-02-14)
Hallman, L. (2006-12-20)
Holmberg, G. (2006-11-29)
Isendor, I. (2006-10-13)
Milani, F. (2006-12-06)
Neck, M. (2006-11-17)
Nyström, C. (2006-11-29)
Rapp, M. (2006-10-06)
Rapp, M. (2006-12-18)
Sundkvist, M. (2006-12-11)
Walter, M. (2006-11-21)
Wickström, I. (2006-10-23)
Appendix

Interview guide
The semi-structured interviews followed this interview guide with on beforehand made questions. Since the thesis spans over the whole range from corporate strategy to implementation of software different categories of professionals were interviewed, both business and IT; developers, managers, architects, owners and administrators. Programmers from both service consumer and service producer side were interviewed, as well as PARI-specialists and non PARI-specialists. Depending on the interviewee’s professional occupation relevant questions were chosen and asked, not necessarily in the same order as presented below.

Battery of questions
The questions served as a starter of discussions concerning a specific topic as well as aiming for specific facts. The interviewee was asked to exemplify their statements and follow-up questions were asked.

What is the overall goal of Handelsbanken? How do you measure it?

What are your strategic guidelines? How are they made concrete?

What is your overall goal? How do you measure it?

How do you link IT-strategy to business strategy?

When you formulate business goals, do they consider opportunities presented by IT-infrastructure?

Which Key Performance Indices do you measure?

Which are your greatest challenges in your profession?

What would the business impact be if those challenges were conquered?

How important is the ability of IT to enable new business opportunities?

In what situation is IT used as an enabler of rationalization? Why?

In what situation is IT used to leverage growth? Why?

What do customers pay for?

What do your business processes look like?

What kind of IT-functionality do you demand? When? For how long?

What are the main drivers for a shared (between BU:s) IT-infrastructure?

What is the life cycle time of business products and services? What is the LCT of the IT-support?

What is the linkage between product/service and IT-support?

How different are IT-support in your domain?
How hard is it to replace current IT-support? What are the most important inhibitors?
What kind of requirements are considered for development of new IT-support or when buying IT-support software from vendors?
What kind of IT-functionality do you buy from vendors?
How much can IT cost? What is the situation in 5 years?
How important is the cost and quality of IT?
How much IT-functionality is bought from vendors? How does the trend look like?
How often does the requirements on IT functionality change? What kind of changes?
How soon does the change need to be implemented?
How well do business managers know what IT-functionality to order?
How much resources are located to IT-functionality? What is the trend?
Is IT an inhibitor to how business is conducted and what business you are in? Why?
In what way can you create competitive advantage by an IT-infrastructure realized through common shared components?
If you could ask for anything IT related, what would it be?
What are the challenges of developing IT-infrastructure within business projects?
What are the challenges of developing IT-infrastructure in individual projects?
What form is the best for developing IT-infrastructure?
What kind of projects is your budget for?
What were the priorities in your project?
Describe the context of the system/project?
Who will use the IT-functionality?
What kind of logic is included in the in the IT-functionality?
In what technology is the IT-functionality realized?
What consideration is taken for data stability?
Is the functionality of your system dependent on or affected by other systems using PARI-service?
Does your system relay on that the provided are accurate? Why?
What are the major aspects of efficiency? How important are they?
How will the level of abstraction affect the amount of CPU-time of the service consumed by the service provider and consumer?
Do you appreciate PARI? Why?
What do you expect from PARI?
Do you have a need to express requirements on PARI differently than today? Why?
Do you have a need to use business world concepts for service expression? Why?

Is PARI easy to comprehend? Why?

Are the operations of PARI easy to comprehend? Why?

Is PARI self-descriptive in the eyes of business users/project purchasers or in the eyes of service consumer developer?

What would the consequence be of self-descriptive services?

Should the terminology of HBOM be altered? If so how and why?

How many operations are invoked in each service call?

Do you need other operations than the currently available?

What information (data) do you need from PARI?

What are your thought on the PARI-manual?

Do you have a need for the possibility to use business world concepts for service expression? Why?

What is the possibility to capture business semantics for a business process activity within an operation?

How do you like the service/operations grouping in PARI? Is it logical? Would you prefer another classification?

How does the PARI-service accessing interface affect the design of your system?

Have you had to develop logic that accordingly to you should be offered by PARI?

Have you considered or do you store local copies of party data?

How will the level of abstraction affect the ability to modify and extend services for service providers?

How will the level of abstraction affect the ability for service consumer developers to modify and extend their systems?

How could PARI be altered to make your system development more efficient? In what way would it be more efficient?

How could PARI be altered to make your system administration more efficient? In what way would it be more efficient?

How will the overall complexity be affected by different coupling between services/systems?

How many dependencies does your system/service have on other systems/services?

How many dependencies do other systems/services have on your system/service?

What would the effect be of changing the number of dependencies?

How difficult is it to interchange services? Why?

What affects the connection pattern for services and operations?
What affects the connection scheme for individual service calls?

How common is to reuse functionality already developed within another system/project?

Is the service possible to invoke from different platforms? Why and how?

How will the level of abstraction affect possibility to develop reusable services?

Does your system/service use other systems/services made alone for you?

How do you know if there is a general need for the functionality you develop?

What is done in order to guarantee that development of IT-functionality considers the whole organization and not just single project interests (cheapest/quickest/easiest)?

How dependent is reusability of functionality on the development process? Why?

How much extra time and effort would it take to make a service general/generic?

What kind of risk is involved in reusing functionality developed for other systems/projects?

How is maintenance and administration affected by using services instead of one-to-one connections?

What affects the change/fault propagation in the architecture?

What affects the work needed for both service providers and service consumer developers in order to interchange operations?

What affects possible reuse scenarios for the service?

What affects the need of awareness of reuse in developing services?

What affects the effort needed to develop reusable services?

**Example interview guide**

Before each interview the above presented questions were compiled into an interview guide. Below is an example presented of such an interview guide. This is the one put together for the interview with Johansson, A.H.

- What is your role in the corporation?
  - What is your strategic responsibility?
  - What is your operational responsibility?

- What are your goals and visions for Handelsbanken?
  - For IT in the organization?

- How important is IT on the senior management agenda?
  - What is your role when IT is the topic of discussion?

- How do you link IT-strategy to business strategy?
  - How do you reconcile different demands in the senior management group?
• How important is the ability of IT to enable new business opportunities, when senior management discusses IT related topics?

• How important is the cost and quality of IT, when senior management discusses IT related topics?

• In what way can the organization create competitive advantage by an IT-infrastructure realized through common shared components?

• What are the main drivers for a shared (between BU:s) IT-infrastructure?
  
  o How owns and maintenances the shared IT-infrastructure?
  
  o What is the situation in 5 years? In 10 years?

• When business units formulate business goals, do they consider opportunities presented by IT-infrastructure?

• How much can IT cost? What is the situation in 5 years?

• In what situation is IT used as an enabler of rationalization?
  
  o What is the role of IT-infrastructure?

• In what situation is IT used to leverage growth?
  
  o What is the role of IT-infrastructure?

• What are the challenges of developing IT-infrastructure within business projects?

• What are the challenges of developing IT-infrastructure in individual projects?

• What form is the best for developing IT-infrastructure?

• What is done in order to guarantee that BU:s that orders IT-functionality considers the whole organization and not just their own interests (cheapest/quickest/easiest)?

• What kind of projects is your budget for?
  
  o What is the situation in 5 years? In 10 years?
  
  o To what degree should the IT-department initiate development of IT-infrastructure?
  
  o To what degree should the IT-department own and finance IT-infrastructure?
  
  o What requirements on business value are there for IT-infrastructure?