UNIVERSITY OF OSLO
Department of Informatics

The Role of Trust in Global Software Outsourcing Relationships

Cand. scient. thesis

Vegar Imsland

30th July 2003

UNIVERSITAS OSLOENSIS • MDCCCXI •

Preface and acknowledgements

This thesis is a part of my Cand. scient. degree in informatics at the Department of Informatics, University of Oslo. The work started in January 2002, and the thesis was finished in July 2003.

First of all, I would like to thank my advisors Tone Bratteteig and Sundeep Sahay for inspiration and constructive comments throughout the research. Thanks to the staff at ScanSys and RussCo for sharing their thoughts and experiences about Global Software Outsourcing with me. I would also like to acknowledge Statens nærings- og distriktsutviklingsfond for the funding which made it possible to visit Russia. Last, but not least, I want to thank Elin for patience and moral support.

Blindern, July 2003

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last decades, a trend towards the globalization of business—and of software-intensive sectors in particular—has emerged. These circumstances, in addition to new technical solutions, have had a large impact on software development. Because of advantages like cost savings, access to world-class IT professionals, and shortened time-to-market, several companies explore these new opportunities.

Global Software Outsourcing, or *GSO*, can be described as "a relatively long-term relationship between firms based in different countries to enable software development to be carried out primarily off-shore (in the premises of the firm doing the development)" (Sahay, Krishna, and Nicholson 2003, p. 3). Thus, the core of GSO is software development taking place outside the national border of the customer country. The phenomenon has also been labelled Information Systems Outsourcing (Yang and Huang 2000), Offshore Sourcing (Marriot 2003), and Global Software Development (Carmel and Agarwal 2001).

This trend also brings along questions about how to successfully operate across national and cultural boundaries. In this thesis, a better understanding of *trust* is suggested as a way to deal with some of these problems. The motivation for this approach is that trust is seen to have several advantages in GSO relationships; it enables cooperative behavior, reduces conflicts, decreases transaction costs, and promotes effective responses to crises (Rousseau, Sitkin, Burt, and Camerer 1998). Furthermore, it may "promote open exchange of information and 'interorganizational learning"'(Sydow 1998, p. 32), and "act as an obstacle to opportunistic behavior" (Karahannas and Jones 1999, p. 347).

For several reasons, the literature on Norwegian GSO projects is very limited. This thesis applies concepts of trust to a GSO project—the *SalarySystem project*—having a Norwegian customer and a Russian supplier. The companies are given the pseudonyms *ScanSys* and *RussCo* respectively. The findings provided will hopefully contribute to a better understanding of GSO relationships within this particular context[1].

1.1 Objectives

Within the problem area of this thesis described above, this thesis aims at the following:

> **Identify and describe trust in Global Software Outsourcing relationships.**

This objective is based upon the assumption that trust *is* needed in GSO projects. However, this assumption will be a subject of investigation too. When discussing the objective, the goal is to achieve the following:

- Provide a better understanding of trust.
- Use this understanding to suggest ways to perform successful GSO projects.

1.2 Background

GSO as a phenomenon is relatively new, starting in the early 1990s (Kobitzsch, Rombach, and Feldmann 2001). However, its roots go back to hardware outsourcing in the 1960s (Lee, Huynh, Kwok, and Pi 2003). During these years, GSO has become a multibillion industry growing from an annual global IT outsourcing market of $96 billion in 1998 to $151 billion in 2000 (Khan, Currie, Weerakkody, and Desai 2003).

The customer countries involved in GSO worldwide are to a large extent located either in Europe or North America—the United States being the largest. Computer-job-related wages moving outside the latter is estimated to $6.5 billion by 2005 (McLaughlin 2003). Other significant customer countries are Japan, the United Kingdom, and Germany.

The undisputed software supplier is India: “India dominates 80–90 per cent of the total of offshore development revenue worldwide and is expected to be the key leader in offshore outsourcing in the next 5 years” (Khan, Currie, Weerakkody, and Desai 2003, p. 240). Other software supplier countries are shown in table 1.1 on the next page, classified by status in the global market.

[1]The problem of generalization from single interpretive case studies is discussed in section 1.3 on page 9.

Leaders	India
Challengers	Canada, China, Czech Republic, Hungary, Ireland, Israel, Mexico, Northern Ireland, Philippines, Poland, Russia, South Africa
Up and Comers	Belarus, Brazil, Caribbean, Egypt, Estonia, Latvia, Lithuania, New Zealand, Singapore, Ukraine, Venezuela
Beginners	Bangladesh, Cuba, Ghana, Korea, Malaysia, Mauritius, Nepal, Senegal, Sri Lanka, Taiwan, Thailand, Vietnam

Table 1.1: GSO suppliers and status (Gartner 2003).

1.2.1 Norway as a software customer

Historically, the Norwegian software sector has not been very visible in the global domain. There have been pockets of some offshore work being undertaken by Norwegian firms in India and Russia though. The relatively low international presence of Norwegian companies has several reasons. One is the competitive oil and gas industry which has led to a certain degree of neglect by the Norwegian government of other sectors like software (Reve and Jakobsen 2001; Økonomisk Rapport 2003). Moreover, Norway being a small country with respect to population, the IT industry has not been very visible worldwide. The total value of the Norwegian IT services market reached €2.2 billion in 2000 and is expected to grow to approximately €3.7 billion in 2005 (Fujitsu Invia Group 2002).

1.2.2 Russia as a software supplier

Russia became a global software supplier as a result of the fall of the Soviet Union, when lots of highly skilled people previously working in scientific and defense establishments were left jobless (Terekhov 2001). Since then, the growth has been substantial. In 2002, the size of the Russian software services was estimated to \$150–\$250 million (Lane 2003) having an expected growth rate at 40–50% a year (The Russia Journal 2003). The Russian educational system is known for its high technical level—students from the St. Petersburg State University won the "ACM International Collegiate Programming Contest" both in 2000 and 2001 (ICPC 2003). This high educational level is also mirrored in the Russian software companies: "77.4 percent of Russian software companies employ PhDs while in 45.8 percent of these companies, PhDs make up about 10 percent of their staff" (Terekhov 2001, p. 99). However, the

Russian software industry also faces some challenges. The country is known both for its low IT infrastructure capability and lack of English proficiency in the population (Marriot 2003; Pries-Heje, Baskerville, and Hansen 2003). Russia's reputation for unlicensed software is a problem too: "Russian salespeople say they spend 80 percent of their time selling Russia and only 20 percent selling products" (Marriot 2003, p. 6).

Figure 1.1 shows the software piracy rates in different countries which are defined as "the volume of software pirated as a percent of total software installed in each country" (Business Software Association 2002, p. 9). As seen in the figure, software piracy is more common in Russia compared to Norway, Denmark, and the US.

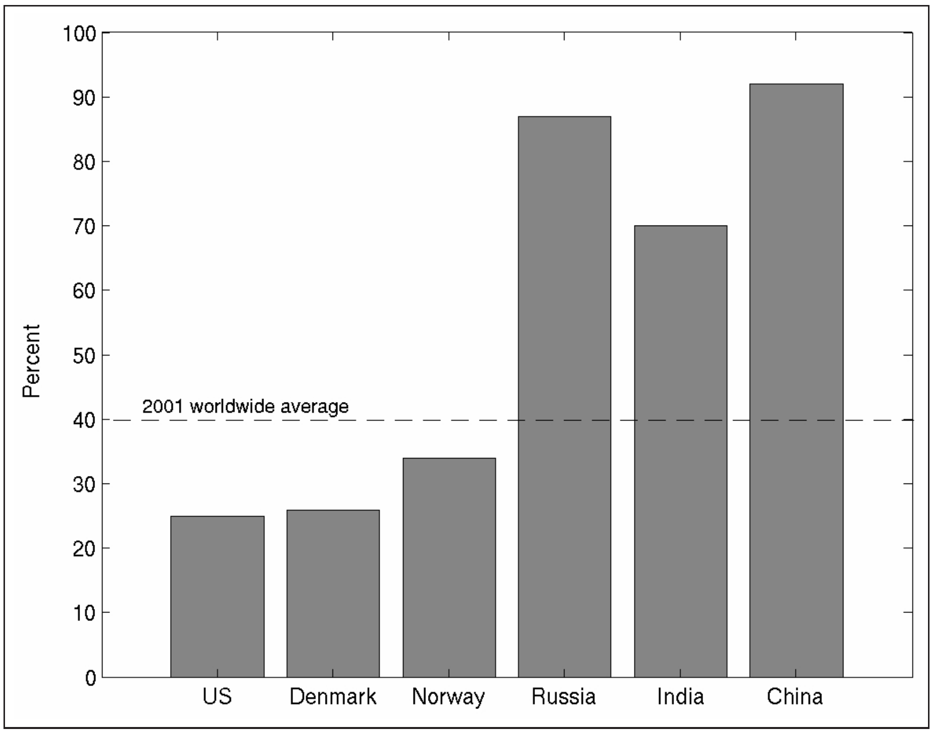


Figure 1.1: Piracy rates in different countries (Business Software Association 2002).

This view is further confirmed by Novell[2] who in its November 1998 Duma testimony said: "80–90% of all the information systems in Russia are based on one or another version of Novell Netware" (McHenry and Malkov 1999, p. 15). Software piracy is a risk in the GSO context because some software suppliers illegally sell the same or a similar product to the system being developed to a third party (The Russia Journal 2002).

[2]See `www.novell.com`.

1.2.3 Strategies for GSO relationships

GSO relationships are taking place using several different organizational relationships. These are briefly outlined below:

Wholly owned subsidiary: Some software supplier firms own facilities overseas to perform analysis and design work at the customer's site, while the rest of the development process is performed in India or Russia (Khan, Currie, Weerakkody, and Desai 2003). Some software customers use this strategy too, either by setting an office offshore—in the country where the software supplier is located—or buying an existing company and make it work together with the already existing organization (Carmel and Agarwal 2001). Using this strategy makes it easier to make the requirements specification since the two parties are located in the same country (Khan, Currie, Weerakkody, and Desai 2003). However, the necessary investments are significant, thus mainly large companies use this strategy. Microsoft, Oracle, Motorola, Ericsson, and General Electric are examples of companies which have set up software development centers in India (Sudan 2000). In ScanSys, the strategy for growth prior to the SalarySystem project, was through acquisitions of existing companies. ScanSys considered this strategy prior to the SalarySystem project, but found the initial investments being too big.

Joint venture: Establishing a joint venture means setting up a new company to run a contract where the customer and the software supplier collaborate. Both parties invest money, so there is a large degree of equity present. Furthermore, this strategy involves risk because of the money that has to be invested. Joint ventures are used by software companies to enter new markets (Carmel and Agarwal 2001; Khan, Currie, Weerakkody, and Desai 2003). However, the strategy might be difficult to use because of language differences (Khan, Currie, Weerakkody, and Desai 2003).

Direct outsourcing to supplier firms: The IT manager or a representative from the customer manages the project from their office, while the development work is carried out offshore (Carmel and Agarwal 2001; Khan, Currie, Weerakkody, and Desai 2003; Gallivan and Oh 1999). This strategy was used in the SalarySystem project. Direct outsourcing to the supplier may be hampered because of difficulties in choosing the appropriate supplier and explaining the requirements of the product because of differences in language and culture (Khan, Currie, Weerakkody, and Desai 2003). This is the most common strategy in GSO projects (Gallivan and Oh 1999).

Multiple Outsourcing Suppliers: The customer company uses several software suppliers to develop the software (Khan, Currie, Weerakkody, and Desai 2003). The advantage of this strategy is the possibility of getting access to the appropriate skills, but it also includes problems of coordinating the project(s) (Gallivan and Oh 1999).

Through a third party: Marketing agents or a third party firm is involved in managing the project (Khan, Currie, Weerakkody, and Desai 2003). Using this strategy, the customer does not have to spend time on coordinating the work. However, the strategy includes the risk of loosing touch with the project which, in turn, may cause the product becoming different compared to what was initially intended (Khan, Currie, Weerakkody, and Desai 2003).

Body shopping: A contract employee is being managed by the customer (Heeks, Krishna, Nicholson, and Sahay 2001; Khan, Currie, Weerakkody, and Desai 2003; Nicholson and Sahay 2001). Using this strategy, the work is taking place on site rather than offshore. This strategy includes low investments for the customer (Khan, Currie, Weerakkody, and Desai 2003).

As described above, GSO projects take place using a number of different strategies associated with strengths and weaknesses. This diversity implies that GSO is a concept consisting of several forms and arrangements.

1.2.4 Accelerators

There are several reasons for the growth of GSO over the last 10–15 years. One reason is the opportunity of reduced costs because of cheap labor (Herbsleb and Moitra 2001; Heeks, Krishna, Nicholson, and Sahay 2001; Marriot 2003; Kobitzsch, Rombach, and Feldmann 2001; Khan, Currie, Weerakkody, and Desai 2003; Carmel 1999; McFarlan and Nolan 1995). Because the salaries of software developers are higher in Northern America and Western Europe compared to other countries, the potential for lower development costs exists. Table 1.2 on the next page shows the basic salary of software developers in different countries. A rough estimate of the corresponding Norwegian salaries—based on NIF (2002)—is \$45 000–\$65 000.

However, GSO relationships, compared to in-house projects[3], include an overhead in relation to communication, knowledge sharing, project management, coordination, and travel costs, so the savings are

[3]In-house projects are performed within the organization as opposed to GSO projects which are performed by another company.

Country	**Base annual Salary (US $)**
China	$4750
India	$5850
Philippines	$6550
Russia	$7500
Indonesia	$12 200
Japan	$44 000
United States	$63 000

Table 1.2: The basic salary of software developers in different countries (Khan, Currie, Weerakkody, and Desai 2003).

smaller than the salary differences in table 1.2 may indicate. The potential for cost savings are significant though: "Industry analysts estimate that hiring programmers outside the US in locales such as India saves about 30 percent in salary costs" (McLaughlin 2003, p. 114).

Another advantage of GSO is the possibility to decrease the skills shortage (Heeks, Krishna, Nicholson, and Sahay 2001; Marriot 2003; Kobitzsch, Rombach, and Feldmann 2001; Carmel 1999). The needed competence may sometimes be unavailable both within the company's organization and inside the customer country itself. Such skills are in many cases available on the global market.

At last, GSO projects sometimes take place because of the need for short development time—enabled by a large pool of available developers (Herbsleb and Moitra 2001; Marriot 2003; Carmel 1999).

There were two main reasons for SalarySystem becoming a GSO project: first, ScanSys needed the software to be developed at a low cost, and second, the company lacked Delphi developers inside its organization. These reasons are further elaborated in chapter 3.

1.2.5 Inhibitors

During the last decade, companies have faced many problems related to GSO projects. Such projects are quite different from projects taking place in-house—a difference which makes such projects difficult to manage. Since the customer and the supplier in GSO projects are sometimes located in different time zones, managing this difference may be problematic (Marriot 2003; Kobitzsch, Rombach, and Feldmann 2001). Although asynchronous communication—taking place using e-mail, online discussion groups, and bug tracking databases—is useful, this kind of communication is considered second best compared to synchronous communication (Carmel and Agarwal 2001). Synchronous communication like telephone calls and video conferencing are important because they enable problems and misunderstandings more easily to be resolved at an early stage. However, the use of such "real time" communication tools requires time zone differences to be less than a normal day at work between the sites of the customer and the supplier. Thus, if the difference is more than, say, eight hours, synchronous communication tools are not useful unless project members from (at least) one of the sites stay at work outside regular office hours.

Geographical distance may also include differences in language and culture between the GSO partners (Marriot 2003; Kobitzsch, Rombach, and Feldmann 2001). Such differences can cause misunderstandings and communication problems. One example of cultural differences that may affect communication, is high and low context communication: "In low context communication, [. . .] the success of the interaction rests primarily on the sender of the message. [. . .] In contrast, high context communication is implicit in nature" (Nance and Strohmaier 1994, p. 117). That is, while low context communication is the spoken words, high context communication sometimes convey more information about the topic being discussed than the spoken words themselves. In Norway low context communication is most common, while Russia is seen to rely on high context communication (St.Amant 2002). In the SalarySystem project, such problems occurred when the Norwegians expected the Russians to be straightforward when reporting the project status, while the Russians expected the Norwegians to "read between the lines".

Also, if the understanding of the problem domain of the project requires extensive knowledge, such knowledge needs to be shared: "[. . .] when a company has specialized for several years in developing control software for a very specific domain—such as for nuclear power plants or cardiac pacemakers—it must usually transfer the knowledge it has gained about the specific control algorithms to the partners" (Kobitzsch, Rombach, and Feldmann 2001, p. 80).

Another problem experienced in GSO projects, is inadequate infrastructure (Marriot 2003; Kobitzsch, Rombach, and Feldmann 2001). The problem is present especially in some of the supplier countries, for instance China, Hungary, and Russia (Marriot 2003). If transport and data transmission is being hampered, such difficulties may cause delays and frustrations in the project.

At last, both the customer and the developer in GSO projects lack full control (Sabherwal 1999; Marriot 2003). Because the GSO partner is located far away, his or her actions are invisible and may be difficult to understand. According to Giddens (1990) uncertainty and lack of control are the prime conditions for trust. That is, if the trustee cannot control the trustor, trust is likely to emerge as a "substitute". As a result, the role of trust in GSO relationships is the focus of this thesis.

1.3 Research approach

In this section I present the research approach used to explore the problem statement of the thesis. One of the research goals is to understand more about GSO as a phenomenon with a focus on issues of trust, and examine how these issues can be managed better with a view to improve the performance of such relationships. Since the focus is on understanding, and to some extent change, an interpretive research approach is chosen.

1.3.1 Interpretive research

Following the classification of Braa and Vidgen (2000), there are two separate research approaches used in IS research: positivist and interpretivist. Positivist research assumes that a phenomenon can be observed objectively. The researcher is observing on the "outside" without influencing the research. Interpretivist research is concerned with interpreting and describing a phenomenon where the researcher tries to understand the insider's point of view, even by becoming an insider in the research.

A further distinction to make in this context is the role of the researcher. Walsham (1995) identifies two roles within the interpretivist approach: the outside observer and the involved researcher. The advantage of being an outside observer is that personnel will be honest because the researcher has no personal interest. On the other had, he or she may not get access to all data. The advantage of being an involved researcher is the possibility of getting access to much data, including sensitive information. The disadvantage is being *too* involved in the field site, at the risk of taking a consultancy role. Having the research goal of this thesis in mind, the involved researcher role was chosen—aiming at understanding the project, and to a lesser degree influencing change.

The specific methods for gaining knowledge using an interpretive research approach include interviews, attendance at meetings, discussions with the people studied, looking at documents, and interpreting the use of tools and artefacts (Klein and Myers 1999). A way of describing this approach is the following: "What we call our data are really our own constructions of other people's constructions of what they and their compatriots are up to" (Walsham 1995, p. 75). Thus, interpretation is the subjective result of all the different ways of trying to understand the project. Because of this, interpretivists need to explicitly say *how* they arrived at their results.

Action case

By focusing on the *intended outcome* of research; change, prediction, and understanding are identified. Figure 1.2 on the facing page from Braa and Vidgen (2000) shows these outcomes and their interconnections. Although the initial focus in the research was on understanding, the following quote shows that I also caused change:

> Regarding our plans for the future, the cooperation with [RussCo] will be based on the [ScanSys] experience and advice from the University of Oslo. We will try to improve and develop this new knowledge.
>
> — *ScanSys company manager*

During the research, the change I caused was found to be a useful way of testing the validity of my interpretations. Thus, without changing the focus of the research, change was seen as a useful way of gaining new knowledge. However, since the change I caused on the research site was more a side-effect than an intended outcome, labelling my field study "action research" would consequently be incorrect. Instead, I choose to follow Braa and Vidgen (2000) in their concept of *action case* which is seen as a "hybrid of understanding and change" (Braa and Vidgen 2000, p. 249).

Reflections

The main advantage of using an interpretive research approach is the possibility of getting access to divine data (Walsham 1995). Such data may, in turn, enable a broad understanding of the research site.

A problem of single interpretive case studies is how to generalize the results from them. The short answer is to generalize the results as tendencies, not predictions (Walsham 1995). In this thesis, the results are presented as such tendencies based on an iterative process of empirical data analysis and literature. Another problem connected to interpretive research is the difficulty of reporting the part one has had in the project (Walsham 1995). Because the researcher works inside the context of the research, he or she will cause varying degrees of change depending on the nature of his or her involvement. For instance, my interpretation of the role of one of the individuals involved in the SalarySystem project, the "Cultural Liaison" described in section 3.3.1 on page 34, was discussed with the "Cultural Liaison" himself. Also, by sending an early version of a report (Imsland, Sahay, and Wartiainen 2003) to ScanSys and RussCo, my presence in the SalarySystem project caused change I cannot separate from the rest of the project. Furthermore, ScanSys sponsored one of

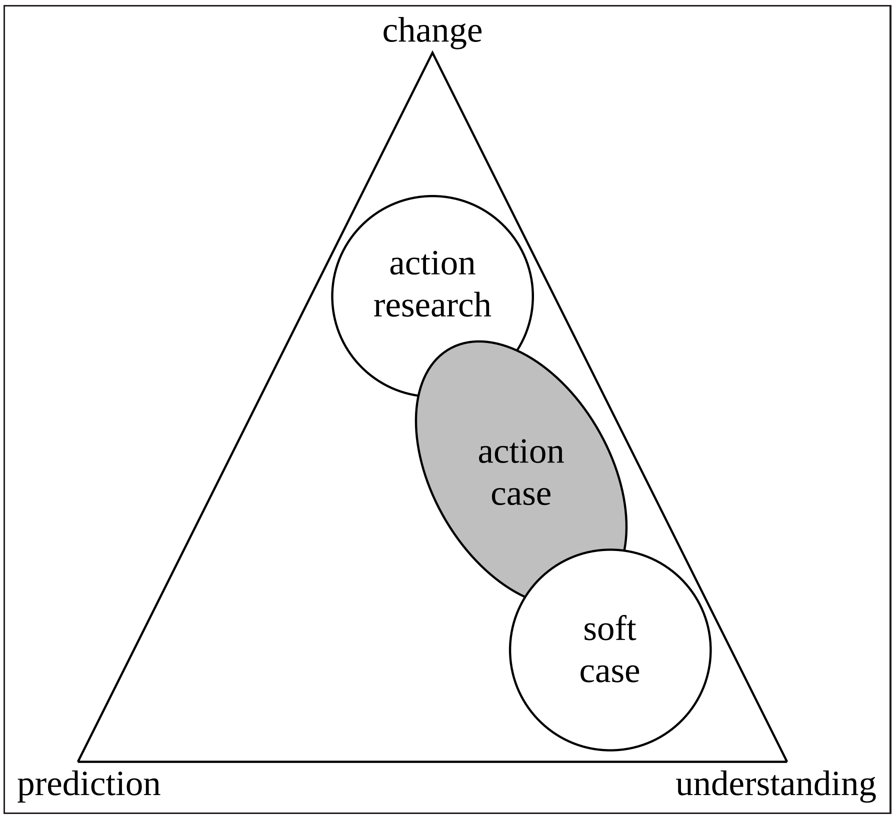


Figure 1.2: Research methods (Braa and Vidgen 2000).

the trips to Russia financially which had the potential of influencing my autonomy in the project. However, I was never instructed in any way what to write, rather the opposite was explicitly encouraged:

> You write whatever you want. I am interested in your opinion from an outside point of view.
>
> — *ScanSys company manager*

These actions—including the feedback and comments I got—increased my understanding of the project. One example is my suggestion of enabling more communication channels between ScanSys and RussCo, where I received the following feedback:

> I do not agree with you. Spaghetti communication will always lead to failure.
>
> — *ScanSys development director*

This feedback made me reconsider my view of what is the most appropriate interorganizational communication model in GSO projects.

1.3.2 What has been done

The study of the ScanSys–RussCo GSO relationship started in the summer of 2002—six months after start of the SalarySystem project—and lasted till the summer of 2003. The data has been collected in several ways including interviews, attendance at meetings, informal discussions when "hanging around" the research site, observation of project members at work, document analysis, reading e-mail transcripts, and examining how to use some of the tools from the SalarySystem project.

In Norway, the research took place mainly at the Norwegian ScanSys headquarters where most of the project staff was located. The support department for SalarySystem at Gran was also visited in addition to the ScanSys subsidiary in Sandnes. The research also included two trips to St. Petersburg, Russia where the RussCo headquarters is located.

The interviews

Most of the interviews were made together with two other people from the University of Oslo; a cand. scient. student and a professor. The interviews took place with one up to three interviewers, and lasted from 30 to 120 minutes. During the research, we conducted 27 interviews with 22 different people. We also attended several meetings concerning the project in different ways. At these meetings there were from three up to ten people in addition to the interviewers. Table 1.3 on the facing page shows the interviewees and their roles in the project. Except from the "President and CEO[4]" of "Indian Supplier Company"—an Indian experienced in GSO—all interviewees were connected to the SalarySystem project in one way or the other. The interviewees were chosen from all hierarchical levels in ScanSys and RussCo, covering as many different roles as possible. By choosing a heterogeneous collection of interviewees, a broader understanding of the project was hopefully achieved. Simultaneously, by interviewing key people of the project several times, in-depth information was found.

In the study we chose a semi-structured interviewing approach which is defined as follows: "Instead of designing a priori a specific set of questions to be asked in a specific order, analysts have various types of questions at their disposal to be used in opportunistic ways, depending on the demands of the situation" (Wood 1997, p. 52). The advantage of such an approach is a flexible interview where interesting issues can be followed

[4]Chief Executive Officer.

Role	**Company**	**# interviews**	**# meetings**
Company Manager	ScanSys	2	4
General Manager	ScanSys		3
Test Coordinator	ScanSys	3	
Developer	ScanSys	1	
Project Manager	ScanSys	1	1
Support Coordinator	ScanSys	1	
Tester III	ScanSys	1	
Development Director	ScanSys	1	3
Technology Director	ScanSys	1	2
Project Leader (on previous GSO projects)	ScanSys	1	
The “Cultural Liaison”	—	1	3
President and CEO	Indian Supplier Company	1	
General Manager	RussCo	3	1
Tester I	RussCo	1	
Tester II	RussCo	1	
Former Project Manager	RussCo	1	
Managing Director	RussCo		1
Project member	RussCo		1
Project Coordinator	RussCo	1	1
The “Delphi Guru”	RussCo	1	
Developer I	RussCo	1	
Developer III	RussCo	1	
Developer IV	RussCo	1	
Developer II	RussCo	1	
Project Manager	RussCo	1	
Sum		**27**	**20**

Table 1.3: Attendance at interviews and meetings.

up by additional questions (Repstad 1998). Because our understanding of the project was limited when the study started, we found it useful to let the interviewee speak about what he or she believed was important in the project. This way we also got a better understanding of what events were necessary to explore in more detail. Only when the interviewee talked about events irrelevant to the project, we found it necessary to interrupt. Our interview guide, with the questions we had at our disposal, can be found in appendix A on page 102. The interview guide includes four kinds of questions:

1. General questions asked initially in the research process.
2. Specific questions asked as the understanding of the project increased.
3. Introducing theoretical concepts.
4. Ad hoc questions.

The first kind of questions was asked in the beginning of the case study. At that time, we needed a general overview of the project in order to pinpoint interesting issues. Later, when our understanding increased, we added questions of the second kind which were more specific with respect to the focus of our research. We also introduced some theoretical concepts (third kind) like “local knowledge”[5] and the different attributes of trust described in section 2.2 on page 20. We did this only when we felt the interviewee had no more to say about an issue unless we introduced our concepts. By using our theoretical ideas in such a way, we also got an impression of how well these ideas fitted with the events they were supposed to describe. At last, the ad hoc questions in the interview guide (fourth kind) are examples of questions we asked spontaneously because we believed they were relevant for our understanding of the project.

When we had identified some issues and episodes we believed was important to the project, we asked other interviewees to comment the same episodes in order to get a better understanding of what happened. One example of such an episode was the removal of the first RussCo project manager described in section 3.4.3 on page 40.

We usually started the interviews by introducing ourselves and told about our roles and intentions in the project. Also, we performed the interviews without recorder. Since we were three interviewers on most of the interviews, we found it unnecessary to take the extra workload of typing the interviews from the recorder. Instead we made notes during the interviews and typed them into a PC as soon as possible after the interviews were made as suggested by Walsham (1995). We also discussed the interviews and filled in missing parts, or cleared up eventual misunderstandings. After typing an interview, we added a part called “personal opinion”. This part was useful because we got more impressions from the interviews than the spoken words. Examples of such impressions include body language, emotions, or if we felt the interviewee did not tell the truth.

[5]The knowledge which was in the heads of people and embedded in work practises.

Literature study

Although GSO is a relatively new phenomenon, there is an abundant number of articles to choose from in scientific databases like ACM[6] and IEEE[7]. Thus, the challenge was to choose the most relevant literature rather than how to find it. Because my area of interest primarily was related trust in GSO project, I was able to choose from a smaller selection. Despite the abundance of literature on GSO in general, close to nothing was found about GSO relationships with Norwegian companies. However, I found that GSO experiences from other countries in Western Europe and North America were useful to some extent.

Literature on trust was also available in large amount. The concept of trust has been found interesting in a number of different disciplines, and because I wanted a broad starting point when I defined trust, I chose literature from disciplines like psychology, sociology, and economics.

The goal of the literature study was to get an appropriate understanding of GSO with a focus on problems and solutions. Thus, the literature study and the analysis of the field data took place as an iterative process as suggested in Walsham (1995). Furthermore, the literature study also served the purpose of making the attributes of trust able to identify trust in the SalarySystem project.

Other sources of data

From the SalarySystem project, I had access to a number of different documents. Some examples include the Software Requirements Specification, a test case, the test plan, two weekly status reports from Russia, e-mail transcripts, and several documents describing SalarySystem. Some of these documents were useful for the historical reconstruction we made since our study started six months after the SalarySystem project was launched. Also, the software analysis documents like the Software Requirements Specification and the test case was important in order to analyze the knowledge sharing between ScanSys and RussCo.

Another way of collecting data from the SalarySystem project was by analyzing the use of *TestTool*—the tool used for managing bugs in the SalarySystem source code. By learning how to use this application, a better understanding of the software development process was achieved. In addition, the use of TestTool was further investigated by observing some of the SalarySystem project members while using the application.

[6]See `portal.acm.org`.
[7]See `ieeexplore.ieee.org`.

1.4 Limitations

This thesis focuses on trust in GSO relationships—two large areas of research. Thus, there is a need to limit the focus in order to fit with the scope of this thesis.

Trust is often discussed in relation to concepts like risk, control, and power. In this thesis, none of these will be the discussed explicitly. Instead, they are discussed when their presence are important for the understanding of trust.

Although GSO provides the empirical background of this thesis, an extensive discussion of the phenomenon will not be made. For instance, GSO relationships exist in a context where culture and language makes communication difficult. These issues will not be discussed in detail. However, issues of communication are addressed by Wartiainen (2003) using the empirical findings from the SalarySystem project too. Furthermore, the technical issues related to software development like methodology and programming will not be emphasized either.

The empirical data has also limited the focus in several ways. The study of the SalarySystem project lasted for approximately one year and included neither the beginning nor the end of the project.

1.5 Outline of the thesis

This chapter has introduced GSO and established the need for trust in such projects followed by a presentation of the research approach. The remainder of this thesis is organized as follows:

Chapter 2 provides the theoretical framework wherein trust is discussed in addition to an evaluation of the suggested attributes of trust.

Chapter 3 describes the SalarySystem project in a chronological way; from the first version of the product to the end of the re-engineering project. The two companies in the project; ScanSys and RussCo, and the product; SalarySystem, are described too.

Chapter 4 provides an analysis where the suggested attributes of trust are applied on the SalarySystem project.

Chapter 5 summarizes the findings from chapter 4. Based on these findings, the understanding of trust and the role of trust in GSO relationships are discussed.

Chapter 6 summarizes the findings from this thesis and discusses directions for future research.

Chapter 2

Trust

In this chapter, the framework which trust is discussed within is presented. Since trust is a complex phenomenon and difficult to identify, seven identifiable *attributes of trust* are suggested. In order to give the attributes explanatory power in the GSO context, some *factors about trust* are considered important for the attributes to address too. First, the factors about trust are presented in addition to how they are addressed by the attributes of trust. Second, the attributes are described in turn. At last, the attributes of trust are evaluated.

2.1 The complexity of trust

This thesis focuses on understanding the role of trust in GSO relationships. In such relationships, establishing trust is suggested to have several advantages. Because companies participating in GSO relationships are geographically separated, several risks appear as a consequence of lack of information about what the partner is doing. This uncertainty has to be replaced by something which decisions can be based upon, and in this thesis trust is seen as a possible solution: "Trust is related to absence in time and space. There would be no need to trust anyone, neither individuals nor abstract systems, if their activities were visible and easy to understand. So the prime condition for trust is lack of full information" (Giddens 1990, p. 33). This relationship between trust and risk is also acknowledged by Das and Teng (2001) and Karahannas and Jones (1999).

Trust as a phenomenon is complex and has many meanings, and no widely acknowledged definition of the term exists. The definitions and conceptualizations about trust are diverging even if they include *some* shared characteristics (Rousseau, Sitkin, Burt, and Camerer 1998). The study of these shared characteristics forms the basis for understanding trust in this thesis; the characteristics are presented as *attributes of* *trust*. The following attributes are identified: *predictability*, *competence*, *structure*, *calculation*, *goodwill*, *knowledge*, and *betrayal*. The selection of these attributes is based on the study of relevant literature with the purpose of making them identifiable in the GSO context. In order to achieve this, there are several factors that have to be considered. These factors are seen to be necessary to be addressed for the attributes of trust to have explanatory power in the GSO context. The connections between the factors and the attributes of trust are depicted in table 2.1 on the facing page. As seen in the table, all the factors are addressed by the attributes. Below is an outline of the factors, and how they are addressed by the attributes of trust.

The notion of trust has been studied by *a number of disciplines*, each emphasizing different aspects: "researchers in different disciplines have viewed trust along different dimensions" (Kim and Prabhakar 2000, p. 538). Economists tend to view trust as calculative, psychologists emphasize the personal attributes, while sociologists consider the institutional properties (Rousseau, Sitkin, Burt, and Camerer 1998). In this thesis, ideas from all these three disciplines are drawn upon because aspects from economics, sociology, and psychology are seen to be relevant in GSO relationships.

In GSO relationships trust is important *within and between* organizations, and is different in these two settings (Rousseau, Sitkin, Burt, and Camerer 1998; Sydow 1998). This thesis looks at trust in both these environments.

Trust has also been found to differ regarding what *organizational level* it is studied. Because trust is different at the individual, group, and institutional level (Rousseau, Sitkin, Burt, and Camerer 1998; Sydow 1998), this thesis will consider trust at different levels too.

Trust has not only to do with relationships between *humans*, but also concerns *systems*. While trust in humans stems from interaction, trust in abstract systems has its source in faith in the correctness of principles (Giddens 1990). Both personal and system related trust is addressed in this thesis.

Trust has been found to *change over time* (Sydow 1998). Usually, three phases are identified: building, stability, and dissolution (Rousseau, Sitkin, Burt, and Camerer 1998; Child 1998). In this thesis, trust will be considered during the whole project—from the formation of the relationship to the end of the project, and how it changes over time.

Trust does not have a constant degree, but is found to have varying *degrees*. The degree of trust in a relationship may vary over time, as well as between different relationships: "It is possible to both over and underinvest in trust, and neither is desirable from either a moral or strategic point of view" (Wicks, Berman, and Jones 1999, p. 99). Because it is possible to trust little or more, trust is said to have a dynamic nature (Rous-

Factors about trust	**Attributes of trust**						
	Predictability	**Competence**	**Structure**	**Calculation**	**Goodwill**	**Knowledge**	**Betrayal**
Multi-disciplinary			Addresses ideas from sociology	Addresses ideas from economics	Addresses ideas from psychology		
Within and between organizations	Implicit in all the attributes of trust						
Multilevel	Implicit in all the attributes of trust						
Personal – System	Trust in systems and personality	Trust in systems and personality	Trust in systems	Trust in systems and personality	Trust in systems and personality	Trust in personality	Trust in systems and personality
Changes over time	Implicit in all the attributes of trust						
Degree	Implicit in all the attributes of trust						
Bidirectional	Implicit in all the attributes of trust						
Socially constructed	Implicit in all the attributes of trust						

Table 2.1: How the attributes address factors about trust.

seau, Sitkin, Burt, and Camerer 1998). Furthermore, it is possible to trust only *some* of the counterpart's motives, but still be skeptical about others (McKnight and Chervany 2001). When it comes to interorganizational trust, this issue is relevant because it is unnecessary to trust *every* aspect of the partner (Sydow 1998).

Another aspect of trust is that it is *bidirectional*. Although there is a trustor and a trustee, trust goes both ways in a relationship. That is, an individual is both a trustee and a trustor simultaneously. In the thesis, this assumption is implicit when considering trust.

At last, trust is *socially constructed* (Giddens 1990; Grey and Garsten 2001), meaning the social context will determine its specific character. Thus, it is different from project to project—an issue being implicit in this thesis.

As described above, the factors about trust are addressed in the attributes to a large extent which should make them applicable to the GSO context.

2.2 Attributes of trust

With the above assumptions in mind, this section presents the seven attributes of trust. A discussion of the attributes will help to provide a better understanding of trust, and develop a conceptual framework to subsequently analyze the case. By conceptualizing trust by attributes, the different attributes are interconnected. As a consequence, it is important to notice that this division is made only for analytical purposes. That is, attempting to make trust identifiable in the case.

2.2.1 Predictability

One aspect related to trust, is about expectations or contingent outcomes of a process. Giddens (1990) looks at trust as what comes out of the faith in predictability. When a person feels he or she can predict another person's behavior, he is more likely to trust the other person. This prediction can also concern organizations and firms which makes it important in GSO projects. However, in cases where behavior is predictable, it is not necessarily the preferred behavior. One example from the context of GSO could be a company considering outsourcing to well-known provider of pirate copied software. In such a case, it would probably be correct to predict that the company would sell your software illegally, as well. Such a behavior is predictable, but not preferred. Thus, predictability is value-neutral (McKnight and Chervany 2001). Under such circumstances, building trust is difficult, of course. Predictability is still an important aspect though, because unpredictability presents a risk in itself (Maguire, Phillips, and Hardy 2001).

Predicting behavior of others, individuals or organizations, is not an easy task. In the beginning of an outsourcing relationship, when two parties know each other only a little, prediction is difficult. One way to increase the ability to predict is to introduce control mechanisms to make behavior predictable. Another way of predicting behavior, is by observing patterns of behavior, and expecting them to continue (Maguire, Phillips, and Hardy 2001). When predicting by observing, the predictability has to come from increased knowledge about the other. In such cases, the trustor can do little but observe. The trustee, on the other hand, must be careful and act in a consistent way and avoid any sort of unexpected behavior. The process of observation is a slow one. It stems from activity taking place over a period of time, so expecting quick results from this approach, will probably fail.

Predictability has to do with the competence of another, as well. McKnight and Chervany (2001) suggest predictability as a way of addressing the feeling that a person has characteristics beneficial to one. Before a company engages in a relationship, it will normally try to find more information about its future partner; its reputation, amount of success in previous projects et cetera.

When collecting information about another; a person or an organization, the goal is to be able to predict their behavior as precisely as possible. But since the companies participating in GSO projects are located at different sites, full information is impossible to get. Thus, the need for trust is magnified in GSO relationships. What is possible though, is to know enough about the other part to be able to predict behavior in a *reasonably* precise way. Figure 2.1 on the next page shows the connection between degrees of information (predictability) and degrees of trust. GSO relationships are separated geographically, so high degrees of information about the partner are difficult to achieve (arrow 3). Furthermore, since trust is more likely to emerge when the partners know each other well, a low degree of information is not wanted (arrow 1). Neither is a low degree of trust (arrow 4) because trust is needed in GSO relationships. At last, a high degree of trust while the predictability is low is not wanted either (arrow 2). Such a situation could be considered blind trust (Giddens 1990). However, there are degrees of such blindness which should be minimized as far as possible. As the figure illustrates, GSO relationships are squeezed between borders which are not wanted and/or unlikely to cross. The gray square suggests where GSO projects most likely and/or most preferably should be.

Brenkert (1998) analyzes the relationship between predictability and trust, arguing that even when predictability in a relationship is present, trust does not necessarily appear as a result. On the other hand, there are situations where predictability is difficult, but one still has trust. This view is visualized in figure 2.1 where the two examples are located at the lower right and upper right corner respectively. In addition, it makes clear that trust and predictability are not the same which implies that they can both exist independently. Still, as a more general rule, if one cannot predict behavior, trust is difficult to build (Brenkert 1998).

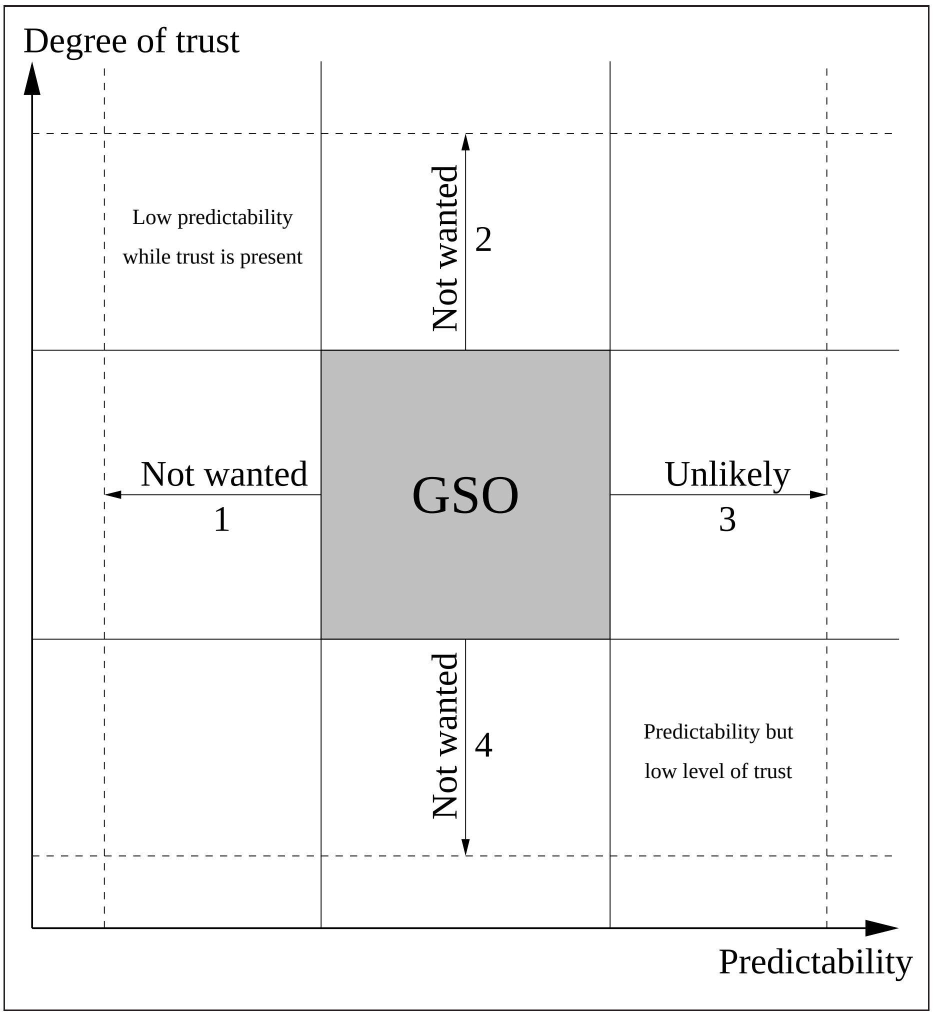


Figure 2.1: The connection between trust and information.

2.2.2 Competence

Competence means having the ability to do for one what one needs done (McKnight and Chervany 2001). It also includes capacity to learn new tasks and technologies (Sahay, Krishna, and Nicholson 2003)[1]. Competence is especially important in the GSO context. In many cases, the key argument for a company to outsource is the lack of competence within its own organization (Carmel 1999). Such competence may be of different kinds though. Technical knowledge is the most obvious in relation to software development, but other resources are important too. Some examples are capital, human resources, physical properties, and market power (Das and Teng 2001).

When being in alliance with a company whose reputation for competence is good, the client will feel more confident about the outcome of the project. Demonstrating competence takes time though, and presupposes close contact between the trustee and the trustor. This contact will again increase the understanding between the partners, and competence trust will be easier to develop. However, in a virtual situation this process develops slower than in a face-to-face situation (Carmel 1999).

Firms possessing different kinds of competence, often try to build a reputation based on these advantages. On the other hand, even a competent partner can act opportunistic (Das and Teng 2001), so basing the project on competence trust alone, is not enough. Sometimes companies try to demonstrate competence they do *not* possess. Such behavior, and its relation with trust, is discussed in section 2.2.7 on page 26.

2.2.3 Structure

The idea of structure as an attribute of trust rests upon the belief that aspects other than personality are important when building trust: "[...] one believes the necessary impersonal structures are in place to enable one to act in anticipation of a successful future endeavor" (Kim and Prabhakar 2000, p. 538). Examples of such structures are written contracts, reporting mechanisms, and rules for response time on written messages (Sabherwal 1999). Also, the use standards like ISO 9001[2] and CMM[3] are considered structural attributes. The key function of these structures is to enable predictability (Maguire, Phillips, and Hardy 2001).

Structures can be used to increase predictability in at least two ways: (1) enable monitoring, and (2) influence the behavior of the partner. The latter function relates to the aspect of control defined as "A regulatory process by which the elements of a system are made more predictable

[1]Sahay, Krishna, and Nicholson (2003) use the term "performance-based trust" defined as "[...] building up confidence with respect to the ability of the partners effectively to carry out the tasks they are supposed to on time and on budget" (Sahay, Krishna, and Nicholson 2003, p. 252). Since this definition to a large extent concerns competence as it is understood above, this thesis interprets the two concepts to be identical.

[2]See www.iso.org.

[3]See www.sei.cmu.edu.

through the establishment of standards in the pursuit of some desired object or state" (Maguire, Phillips, and Hardy 2001, p. 287); (Das and Teng 2001, p. 258).

Considering predictability as an attribute of trust, and structure as a way of achieving this, structure also has a downside with respect to trust. In relationships where the structural mechanisms are either too extensive or too few, trust is difficult to build. While controlling the partner by using structural mechanisms is possible, it may not be the best way to improve performance. Excessive control from one of the organizations can actually hurt performance in an outsourcing relationship because much time has to be spent on reporting and feedback to the controller (Rousseau, Sitkin, Burt, and Camerer 1998). In some relationships, control is used to find ways to identify how the other organization may have hurt the project. Excessive control can also be associated with a perception of a lack of trust. The controlled organization may feel it is not found to be trustworthy, which again can reduce the degree of trust. Excessive faith in trust, while ignoring structural mechanisms, may not be good for the relationship either. If a project is based on the assumption that nobody acts opportunistically, lack of structure can be negative when problems arise. Under such conditions, each organization may try to blame the counterpart, but when there is no formal contract on responsibility, such a situation can reduce trust. Another example is when there are no rules for response time when receiving e-mails. If the sender expects a quick response, and the recipient is slow, the lack of structure may cause irritation which again can hurt trust in the relationship. On the other hand, a way to improve the degree of trust in a relationship is by ensuring that proper structures are present. Because structures enable prediction, but both excessive and too little control can hurt the project performance, a balanced degree of structures is preferred.

2.2.4 Calculation

The idea of calculation as a way of building trust stems from the economic literature, and refers to the assessment of whether the trustee is able to perform an action that is beneficial (Rousseau, Sitkin, Burt, and Camerer 1998; Kim and Prabhakar 2000; Sabherwal 1999). This idea is further elaborated by Castelfranchi and Falcone (2000, p. 1779): "[...] several important examples and natural situations of trust would be excluded without any advantage". This implies that if there is nothing to gain from a relationship, it will not take place. Calculation-based trust appears when the knowledge of the partner is limited: "This form of trust tends to be strongly associated with the early stages of GSA[4] relation-

[4]Global Software Alliance.

ship growth, before other forms of trust (performance and identification) grow" (Sahay, Krishna, and Nicholson 2003, p. 252, footnote added). In the GSO context this kind of trust has to do, for example, with the possibility of signing contracts for future projects (Sabherwal 1999). That is, the software supplier may try to perform well during potential pilot projects aimed at evaluating the competence of the partner.

When a company decides to participate in an outsourcing relationship, the decision also includes several *risks*: the business partner in the relationship may act opportunistically, he or she may not possess the preferred skills, and economic problems may arise. On the other hand, the advantages can be big if the project is a success. When deciding whether to participate in an outsourcing relationship, the company has to compare the potential risks with the possible advantages of the relationship. Thus, risk is not just a negative notion even if the risk of negative events should be minimized as far as possible (Giddens 1999). In our lives there is a balance between avoiding risks and taking them. Inaction is also risky (Giddens 1990), so in every decision, there is an opportunity part along with the possible negative outcome. People take risks in a climate of future opportunities by trying to calculate future risk (Giddens 1999).

A distinction to make is between objective and subjective risk (Das and Teng 2001). Objective risk is based upon the consequences of different alternatives when making a decision. It can sometimes be objectively calculated, for example the probability of winning in a lottery. Subjective risk is the decision makers' estimate of objective risk. Every decision has both an objective and a subjective risk, but because of complexity and lack of information, only the latter is possible to determine. Since risk is associated with uncertainty, the subjective risk is an estimate instead of a prediction (Giddens 1990; Beck 2000).

2.2.5 Goodwill

Goodwill as a source of trust refers to a general tendency to trust others (Kramer 1999; Patrick 2002), and has its origin in the psychology discipline which distinguishes individuals with high and low trusting personalities (Sydow 1998). Disposition to trust is important because it is an antecedent for several other kinds of trust (McKnight and Chervany 2001). Because humans have different experiences regarding their relations to others, these shape the personality with respect to how likely the individual is to develop trust: "Trust develops during childhood as infant seeks and receives help from his benevolent care givers, resulting in a general tendency to trust others" (Kim and Prabhakar 2000, p. 538). This goodwill may concern any attribute of the other, for example personal characteristics like honesty and benevolence, but also more general attributes like competence and predictability (McKnight and Chervany

2001).

The expectation that others have good intentions with their actions may also be relevant on other levels than the individual. Even if trust in organizations, as opposed to trust in individuals, concerns faith in the correctness of principles more than interaction (Sydow 1998; Giddens 1990), such trust is also dependent on a general goodwill between the organizations. Moreover, Sydow (1998) suggests that powerful organizations are more willing to trust then weak ones, and thereby acknowledging a difference in disposition to trust on the organizational level.

2.2.6 Knowledge

Knowledge about the partner(s) in a relationship is seen to be important when building trust: “trust between two or more interdependent actors thickens or thins as a function of their cumulative interaction” (Kramer 1999, p. 575). The most important outcome of such interaction is predictability (see section 2.2.1 on page 20) (Sahay, Krishna, and Nicholson 2003).

It is important to notice that gaining rich knowledge about the partner is difficult to achieve, and emerges only after longer-term interaction (Rousseau, Sitkin, Burt, and Camerer 1998). However, Hertzum (2002) identifies four ways of building knowledge-based trust: *first-hand experience*, *reputation*, *surface attributes* (e.g. language, clothes and so on), and *stereotypes*. Since members of GSO projects are separated geographically, these ways of building trust must be supported by bringing together key personnel. Another way is by launching pilot projects (Sabherwal 1999), and closely monitoring its progress.

In the beginning of GSO projects, when the knowledge about the other organization is limited, Sahay, Krishna, and Nicholson (2003, p. 251) suggest the following solution: “Various middlemen, often experienced, well-connected ex-employees can help facilitate and foster relationships or set up and incubate offshore subsidiaries”. Such people are able to establish important personal relations between the organizations, which is important for sharing knowledge.

2.2.7 Betrayal

Since trust is needed when there is an inability to monitor the action of the trustee—which is the case in GSO projects—means a potential for risk. One of these risks is the opportunity for violations of the trust. This is called betrayal, and is defined as “a voluntary violation of mutually known pivotal expectations of the trustor by the trusted party (trustee), which has the potential to threaten the well-being of the trustor” (Elangovan and Shapiro 1998, p. 548). This definition limits betrayal only to consider intentional acts, and hence, excluding accidents, mistakes, and bad luck. Examples of betrayal are theft, lying, failing to return a phone call, breach of contract, and broken promises. Even if these examples are trivial, their consequences may be significant: "negative (trust-destroying) events are more visible and noticeable than positive (trust-building) events" (Kramer 1999, p. 593).

Another aspect of betrayal is the motives for such a violation. Elangovan and Shapiro (1998) identify several reasons for such behavior to take place: availability of alternate trustors, the recognition of low penalties, the recognition of large benefit, and, at last, a calculative aspect as described in section 2.2.4 on page 24.

Although betrayal to a large extent is associated with trust-breaking behavior, it can also *build* trust. Because behavior can be predictable but not preferred (see section 2.2.1 on page 20), *not* betraying can build trust in (already established) GSO relationships. That is, if the trustee expects to be betrayed, not betraying can build trust in the trustor.

2.3 Evaluation of the attributes

The following attributes of trust have been identified: predictability, competence, structure, calculation, goodwill, knowledge, and betrayal. In order to make the attributes of trust identifiable in the GSO context, they have to be mutually exclusive. That is, the overlaps between them should be minimized as far as possible. However, conceptualizing trust by attributes is made only for analytical purposes. Thus, they are all interconnected to some extent. From the above descriptions, two overlaps were found to be of larger extent than the rest.

First, predictability and calculation are seen to be quite similar. When individuals are assessing whether a relationship will be advantageous, this action includes an element of predicting which connects calculation and predictability. However, while predictability is a source of trust itself, calculation-based trust is seen as what comes out of the prediction of future events. Thus, the two attributes concern different issues of trust although they overlap to some extent.

Second, competence and knowledge are connected since they both concern knowledge about characteristics of the partner. Thus, knowledge about the partner's competence is still knowledge. However, competence is especially important in GSO projects, so competence as an attribute of trust is found to be useful in addition to knowledge in order to identify trust in GSO relationships.

2.3.1 Summary

As described above, two overlaps between the attributes of trust are found. However, despite these overlaps, all the attributes are found to be necessary in order to identify important aspects of trust in the GSO context. Thus, removing any of them is seen to make trust less identifiable. As a result, the attributes will all be used to identify trust in the SalarySystem project.

Furthermore, trust was seen to be multidisciplinary, present within *and* between organizations, multilevelled, related to humans *and* systems, changing over time, having varying degrees, being bidirectional, and socially constructed. Since the attributes of trust address all these factors which were found to be important for the attributes of trust to have explanatory power in the GSO context, the factors are implicit when applying the attributes to the findings from the SalarySystem project. The following attributes of trust are suggested:

Predictability: Observing characteristics about the partner, and expecting these to continue, is a source of trust. However, in cases where predictability is possible, the observed characteristics are not necessarily the preferred ones.

Competence: Competence of different kinds is important in order to perform successful GSO projects. Such skills might be difficult to demonstrate because of geographical distance.

Structure: Structural mechanisms include standards, rules for response time, reporting mechanisms, and written messages. The key function of these structures is to enable predictability.

Calculation: Calculation means assessing whether a relationship will be advantageous which includes both associated risks and opportunities.

Goodwill: Goodwill as a source of trust refers to a general tendency to trust others. Individuals are different with respect to how likely they will develop trust.

Knowledge: Long-time interaction increases the knowledge about the partner. This knowledge makes it easier to predict the future.

Betrayal: Opportunistic behavior which threatens the well-being of the trustor can break trust. If the trustee expects to be betrayed, *not* betraying can build trust in the trustor.

Chapter 3

The SalarySystem project

In this chapter, the history of the SalarySystem project is presented in a chronological way. Some incidents which took place prior to the project are considered important for the understanding too, so the project description starts by introducing these before the rest of the project is described. Since this thesis discusses issues of trust, the presentation of the history is focused on trust too. That is, only events considered relevant to discuss in relation to trust are mentioned. This chapter begins by introducing the two companies in the project and the product being outsourced.

3.1 The companies

The SalarySystem project took place between a Norwegian customer, ScanSys, and a Russian software developer, RussCo. The two companies are described below.

3.1.1 ScanSys

ScanSys is a European-based multinational IT company specialized in consulting, employing 13 000 people in more than 20 countries. The SalarySystem project included only the Norwegian department though. The company develops software solutions for customers, and having annual net sales of €1.3 billion (in 2002), it is a leading supplier of high value-added IT services in Europe. In Norway ScanSys has more than 900 employees, but only around 15 people were directly involved in the ScanSys–RussCo relationship.

3.1.2 RussCo

RussCo is a software company located in St. Petersburg, Russia. The company was founded in 1993, and had in 2003 110 employees consisting of 74 software developers, 19 software testers, and 17 administration personnel. Most of RussCo's customers are from the United States and Europe in addition to some from South Africa and Australia. RussCo is a member of a St. Petersburg-based consortium of information technology companies aiming at establishing contact with foreign customers.

3.2 The product being outsourced

The computer software *SalarySystem* is an off-the-shelf system for salary and human resources management. SalarySystem is designed for small and middle-sized companies and has 11 000 customers in Norway which represents a 70% share of the domestic market. The system is used for several tasks including salary calculation, accounting, and reporting. Examples of data stored in the system include information about accounts, employees, and salary types. The system can generate reports from these data—either via report templates or defined by the user.

The Norwegian tax and salary rules are deeply embedded in SalarySystem, which makes it quite difficult to use even for experienced users. As an example, a user who calculates the tax of the employees needs knowledge about the Norwegian tax and salary rules in order to perform such an operation. Because of this, ScanSys has developed a support department for SalarySystem users. This department receives more than 2000 phone calls on busy days—an important factor for the success of the software:

> [ScanSys] is not selling a program, but a product. Support is important; 50% of the value of the product.
>
> — *ScanSys developer*

The system dates back to 1987, but the ScanSys management found the version of 2001 to be very difficult to maintain for three reasons. Firstly, because the Norwegian tax and salary rules are changed at least once a year, SalarySystem has to be changed constantly in order to be up-to-date. Over the years, changes have made the quality of the source code bad, so ScanSys saw the need for a better-structured source code. Secondly, when the first Delphi version of SalarySystem was written, the developers themselves learned Delphi during the development phase which made the source code unprofessional from the beginning. Thirdly, in 1999, two years before the SalarySystem project started, all the developers who had been involved in the development of SalarySystem left ScanSys, leaving behind little documentation:

> A big issue in [ScanSys] is the lack of documentation.
>
> — *ScanSys company manager*

So to make the required changes, the new developers had to read the source code in order to understand the system. This process was slow which was seen as a problem as it was difficult to respond effectively to the continuous changes.

When ScanSys realized that SalarySystem needed a better-structured and more flexible source code, they also wanted the programming language to be changed from Delphi 3.0 to Delphi 6.0. But still, because of the strong brand name of SalarySystem and high customer satisfaction, ScanSys wanted to keep the functionality of the system unchanged. Such an update of existing software is called *re-engineering* which is defined as restructuring or rewriting of parts or whole inherited systems *without changing functionality* (Sommerville 2001). Another reason for the re-engineering was the database of the system. This database was licensed which was seen as an unnecessary expense by ScanSys considering other databases were available free of charge. Later in the project, these licenses had to be extended for one more year because of the delay in the project. This resulted in a $77 000 additional expenses for ScanSys. In addition, the existing database was seen to be slow and unreliable.

For several reasons, the SalarySystem software was distributed on CDs sent by post. The cost of distributing the CDs to all the SalarySystem customers was 500 000 NOK, which meant that the re-engineered product *had* to be free of bugs before it was distributed. This constraint was difficult to handle in the project and led to further delay because extensive testing had to be performed.

3.3 Pre-project history: initiating the GSO

When ScanSys found that a re-engineering of SalarySystem was necessary, they made estimates of how much work it would be to upgrade the system internally in ScanSys, which was approximately three years of work for one person. The ScanSys general management believed this was costly, and decided not to develop internally. At that time, ScanSys also lacked internal developers having the right skills to perform the project. For these two reasons, the company manager decided to explore other opportunities, and GSO was found to be a possible solution.

First the ScanSys staff considered different countries. For US companies, India is the number one country to outsource to. 67.7% of the Indian software export goes to the US (Nasscom 2003). However, prior to the SalarySystem project, ScanSys had experience from a small GSO project with India, and they found that time difference and geographical and cultural distance hampered the project. For these reasons Russia was chosen instead. Carmel and Agarwal (2001) also identify advantages of such *near shore* outsourcing. One such advantage is reduced response time when using asynchronous communication tools such as e-mail and ICQ. By the end of the project, ScanSys found that Russia had been a better experience than the project with India despite the problems that occurred. The main advantages were seen to be the close cultural and geographical distance—the European mindset of the Russians was easier to relate to for the Norwegians as compared to their prior Indian experience.

After Russia was chosen, the Norwegians went to Russia, and via the consortium of information technology companies located in the St. Petersburg area, they had a number of companies recommended. They visited some of them:

> We looked at six or seven companies, and I can tell you there is a big difference between them both in terms of business experience and leadership. The premises are often very poor, the surroundings horrible, and the state of infrastructure bad. The quality of their theoretical knowledge on the other hand is high.
>
> — *The "Cultural Liaison"*

Since ScanSys in Norway is quite small, they looked for a company of similar size and found RussCo to fit this description. Another reason for the choice of RussCo was that the company was more "Western" compared to the other companies they visited:

> Some of the other companies were very Russian, like the ones you see on TV. They had a technical focus, and the key people spoke bad English.
>
> — *ScanSys company manager*

Also, RussCo was seen to be less hierarchical than the other companies they visited—a difference which was considered positive by the ScanSys staff:

> The other companies we looked at had one person on top in a hierarchy. This is similar to how things were during the Soviet years, when all decisions had to be made at the top level. When we visited [RussCo], we were introduced not only to the top manager, but also to people of lower rank.
>
> — *ScanSys company manager*

Altogether, these factors made the ScanSys staff feel good about RussCo as a future business partner. In the words of a senior ScanSys manager:

> This decision was based on a gut feeling.
>
> — *ScanSys company manager*

This somehow intangible but strong feeling was also described by a ScanSys consultant who helped them establish the RussCo relationship:

> Was it the smell, the look, the way they talked, the way they listened, or the feeling of mutual understanding?
>
> — *The "Cultural Liaison"*

After RussCo was chosen, a pilot project was launched. This project was divided into smaller phases which could be stopped at any time. The idea was to try and see if the relationship worked out well.

From the RussCo side, a potential cooperation with ScanSys was seen to have several advantages. For some time, RussCo had tried to get into the Scandinavian market. Because most of RussCo's customers were located in the US, the company management wanted to increase the number of countries to cooperate with as a strategy to manage risk, especially given the slowing down of the North American market from 2000 onwards. Also, Scandinavian customers were seen to pay better than North American clients. This strategy was seen as a way of being less vulnerable of economic ups and downs:

> You do not put all your eggs in the same basket.
>
> — *RussCo general manager*

Because of the wish to engage in a business relationship with ScanSys, RussCo tried their best to show their competence and company culture to the Norwegians.

3.3.1 The "Cultural Liaison"

In the SalarySystem project, ScanSys used a person who fits the characteristics of a *cultural liaison*: "The cultural liaison might be a person who travels back and forth between the key stakeholder sites. The liaison's informal role is to facilitate the cultural, linguistic, and organizational flow of communication and to bridge cultures, mediate conflicts, and resolve cultural miscommunications" (Carmel and Agarwal 2001, p. 27). This person described his role in the project in the following way:

> I have only been a cultural bridge trying to create the best atmosphere.
>
> — *The "Cultural Liaison"*

The "Cultural Liaison" of the SalarySystem project was a Norwegian with a Russian wife. He had worked in the computer industry since 1969, and had extensive knowledge about Russia. He also spoke Russian. After ScanSys decided to explore the opportunities of GSO, he was hired as a consultant aiming to help them establish the contact with Russian software companies and to understand the Russian way of doing business. During the initial stages, the "Cultural Liaison" attended meetings in Russia to help the ScanSys company manager act correctly with the Russians:

> [ScanSys] hired [the "Cultural Liaison"] as a "bodyguard" and consultant. He talked about everything, and I think [ScanSys] used him to tell them "who is telling the truth".
>
> — *RussCo general manager*

The crucial role of the "Cultural Liaison" in establishing and maintaining the relationship is confirmed by the ScanSys company manager:

> Without a local guide I would never have continued to try to establish a partnership in Russia.
>
> — *ScanSys company manager*

3.4 History of the SalarySystem project

In this section, the project history is described. To ease the understanding of the project, the history is divided into five parts: the first meeting in Norway, the estimation process, the near-breakdown, the recovery phase, and interdependency. This division is made only for analytical purposes because the events described sometimes took place in several phases. These phases are schematically shown in figure 3.1 and discussed below.

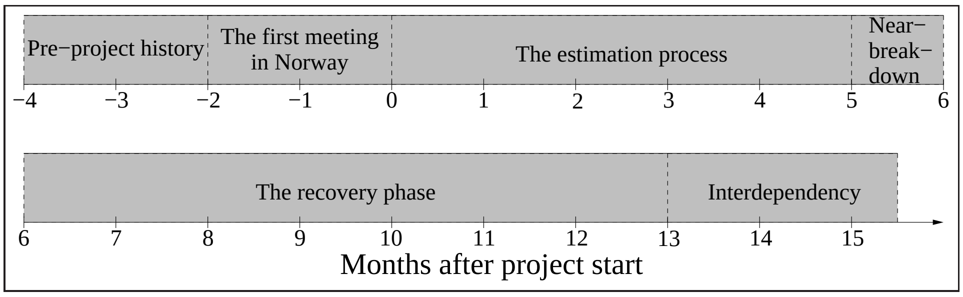


Figure 3.1: The SalarySystem project phases.

3.4.1 The first meeting in Norway

After RussCo was chosen as the outsourcing partner, two RussCo staff came to Norway; the project manager and a developer. The latter was discussed as a very clever developer referred to as the "Delphi Guru" by his RussCo colleagues. From ScanSys, the project manager and a Russian-speaking developer participated.

The aim of the meeting, initiated by ScanSys, was to transfer an understanding of the domain knowledge of SalarySystem as well as learning to know each other better:

> [RussCo] appointed a project leader who came to Norway for one week to go through the specifications and learn more about our organization. Also, [the ScanSys staff] should evaluate the degree of knowledge and experience of the RussCo employees.
>
> — *The "Cultural Liaison"*

The Norwegian project manager was initially skeptical about the decision to outsource SalarySystem—he found it hard to believe that an external development team would be able to perform a successful project. As the meeting continued, he changed his mind:

> They were very persuasive, so I was convinced that they could do the job. What persuaded me was that the Russians came up with the same solutions as we wanted. I went from being

> negative to become positive towards cooperating with [RussCo].
>
> — *ScanSys project manager*

The focus of the discussions was primarily technical, considering the database structure, when to use Delphi standard components et cetera. They looked at the documents describing the new system that the project manager had made, and the Russians had the details explained. These details were explained by the ScanSys Russian-speaking developer—in Russian. Because of this, the ScanSys project manager was not actively involved in the discussion which further amplified the technical approach of the discussions. On the other hand, the use of the Russian language made the technical details easier to understand for the Russians.

The documentation about SalarySystem was to a large extent written in Norwegian. Even the source code had comments, class names, variable names, database fields, and tables written in Norwegian. During the meeting, the decision was made to use Norwegian in the source code of the new system too. The reason for this decision was to maintain consistency between the old and new versions of SalarySystem. Because the two versions would be used simultaneously for a period, this decision was found to be necessary. As a result of these issues, the amount of documentation the Russians got was necessarily limited and difficult to understand. The information they had on the business logic was limited too. After the meeting the Russians brought the 15 Mb source code, an installation guide, the SalarySystem manual, a picture of the database structure drawn from the whiteboard drawn by one of the ScanSys staff, and a running version of the system back to Russia. Most of this was written in Norwegian.

Initially in the project, the ScanSys staff was aware of the business logic encapsulated into SalarySystem. For one part of the system, the calculation engine, the need for understanding the business logic is crucial. This part calculates taxes, salaries et cetera using the Norwegian tax and salary rules. Because of this need for domain knowledge, the decision was made to develop this part of the system in Norway. Furthermore, ScanSys assumed that the rest of the system would be less dependent on the business logic knowledge, an assumption subsequently proved wrong. During the autumn of 2001, RussCo spent a lot of time trying to understand SalarySystem from the information they got. It proved to be difficult to find out all about the business logic because they had little information, and what they had was written Norwegian. They prepared questions and tried to create prototypes, but they could not see the business logic. If the Russians had questions, they collected them and sent an e-mail to Norway. They got some more information this way, but sometimes the Norwegians did not have the answer either since the developers in charge had not been involved in the development of the system.

The starting point of the project was limited to a technical focus, and that too was rather thin in the absence of any formal documentation. This problem was compounded by the fact that when the project manager returned to Russia after the meeting in Norway, he spent only about 10% of his time on the SalarySystem project. The rest of his time was spent on other projects that RussCo was involved in with other customers. The "Delphi Guru" was also shortly after the meeting moved by RussCo to another project, and was for the next six months or so not involved with SalarySystem. The little domain knowledge that had been transferred to RussCo through the meeting in Norway was thus further dwindled by the reassignment of these two key project staff—something that was not explicitly known to ScanSys.

An implication of the poor knowledge transfer was that one of the Russian developers spent one month working on a technical solution that turned out to be wrong. The misunderstanding concerned whether the interface of the system should be stored in the database, or generated from a common GUI[1] framework. Unfortunately, she worked on the wrong solution for one month before this misunderstanding was discovered because no one was there to clarify her work.

Once the RussCo general management realized that the project was running into problems, more people were assigned. The first two months of the project the project team consisted of two developers in addition to the project manager. Two months later they had grown to five people before they ended up at ten developers six months after the project start. But even if these adjustments were made, the performance of the project was low.

3.4.2 The estimation process

After the meeting in Norway, the Russians started to make estimates about the re-engineering of SalarySystem. RussCo estimated to finish the project in six months, using 642 person-days (see figure 3.2 on page 39). 14 months after project start, when the project was about to be finished, RussCo had used 3600 person-days. Since the project estimates were about six times under the actual amount of work, it suggests the estimation process was problematic. However, the RussCo estimates were also confirmed by ScanSys who earlier had made approximately the same estimates internally. But because ScanSys wanted RussCo to make estim-

[1] Graphical User Interface.

ates independently of the ScanSys estimates, ScanSys did not give them to RussCo during the first meeting in Norway. Later, the RussCo general manager said:

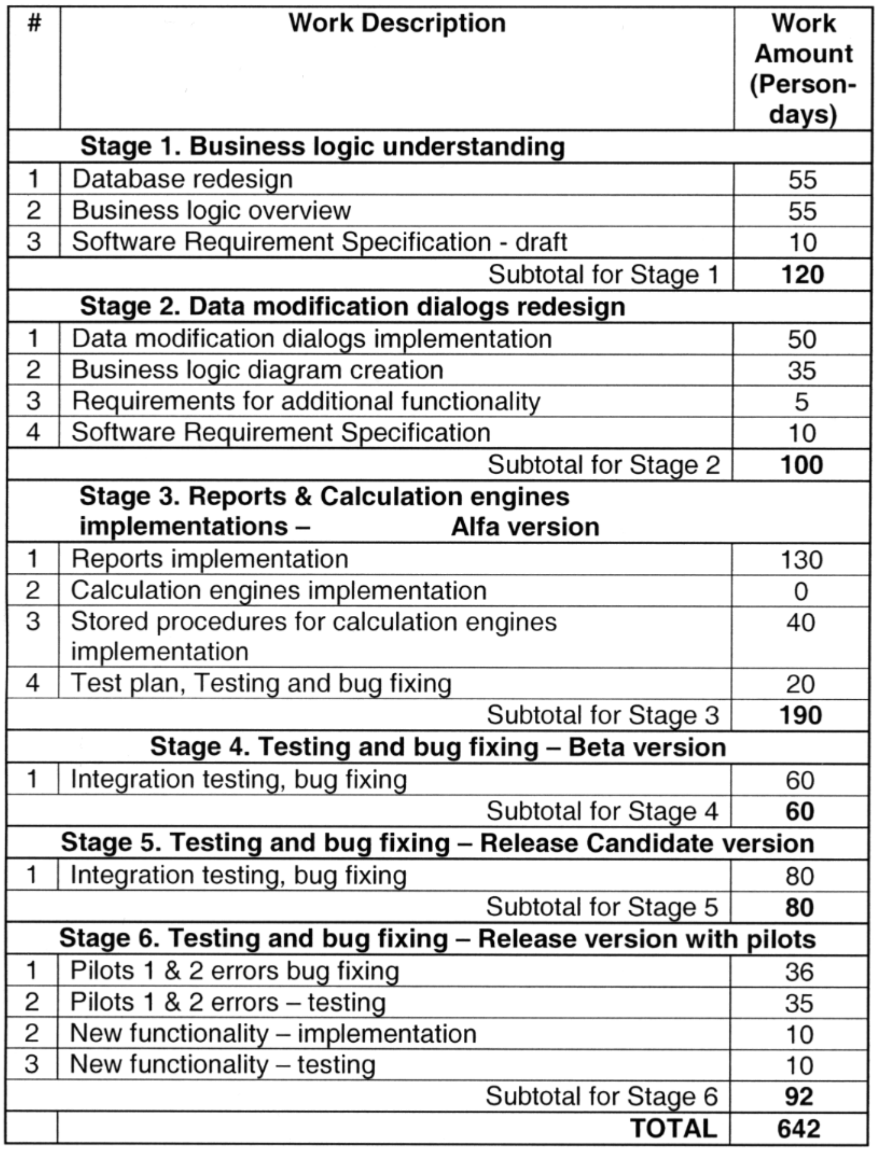
> We may be stupid, but both sides made bad estimates.
>
> — *RussCo general manager*

Despite the limited understanding of the system, the Russians started making estimates about the amount of work to re-engineer the system. Firstly, an attempt was made to estimate the system based on "lines of code". This method, described by Jørgensen (2001), is to prepare estimates based on lines of source code. There are around 300 000 lines of source code in the old SalarySystem, but the estimate was seen to be too high and abandoned. RussCo found this approach inappropriate because they found that since parts of the old code were not being used, the re-engineered source code would consist of less program lines. Also, since change of the database was one of the key issues of the project, the RussCo analysts did not assume the number of source code lines would be comparable in the old and the new system. Later, the following estimation process was adopted: First the project members looked at the source code and the running version of the system, and tried to trace the structure and functionality of the system. Then the Russian project manager asked the project members how much time they needed to develop different parts of the system like forms, tables and so on. Afterwards they counted these items, summed up, and assumed that each of these items would take a given number of days to develop. At last, they estimated 1/3rd in addition for testing. These estimates were sent to Oslo where they were compared with the ScanSys estimates, and it was concluded by the project manager that the estimates were generally okay.

The RussCo estimates of work to complete the SalarySystem project are shown in figure 3.2. These estimates, divided into six stages, are mostly concerned with system implementation. System analysis—including understanding of the domain knowledge—is emphasized less. However, the estimate called "Business logic overview" in stage 1 concerns analysis as well as "Software Requirement Specification – draft" and "Software Requirement Specification" from stage 2. Altogether, the three estimates add up to 75 person-days of analysis out of a total of 642 person-days for the whole project. Consequently, since only these parts of the estimates concerned system analysis, the 75 person-days would have to include both language training and understanding the business logic of the system. However, these two parts of the project turned out to be very time-consuming:

All additional time was spent trying to understand the business logic of the system. The business logic was not understood, and we did not see the complexity of the project. We did not understand the usage of [SalarySystem]. No time was estimated for translation.

— RussCo developer I



#	Work Description	Work Amount (Person-days)
	Stage 1. Business logic understanding	
1	Database redesign	55
2	Business logic overview	55
3	Software Requirement Specification - draft	10
	Subtotal for Stage 1	**120**
	Stage 2. Data modification dialogs redesign	
1	Data modification dialogs implementation	50
2	Business logic diagram creation	35
3	Requirements for additional functionality	5
4	Software Requirement Specification	10
	Subtotal for Stage 2	**100**
	Stage 3. Reports & Calculation engines implementations – Alfa version	
1	Reports implementation	130
2	Calculation engines implementation	0
3	Stored procedures for calculation engines implementation	40
4	Test plan, Testing and bug fixing	20
	Subtotal for Stage 3	**190**
	Stage 4. Testing and bug fixing – Beta version	
1	Integration testing, bug fixing	60
	Subtotal for Stage 4	**60**
	Stage 5. Testing and bug fixing – Release Candidate version	
1	Integration testing, bug fixing	80
	Subtotal for Stage 5	**80**
	Stage 6. Testing and bug fixing – Release version with pilots	
1	Pilots 1 & 2 errors bug fixing	36
2	Pilots 1 & 2 errors – testing	35
2	New functionality – implementation	10
3	New functionality – testing	10
	Subtotal for Stage 6	**92**
	TOTAL	**642**

Figure 3.2: The RussCo estimates.

Another reason for the wrong estimates had to do with the contract of the project. This contract was signed *after* the estimates were made because it was supposed to be based on these. That is, the price ScanSys was supposed to pay for the re-engineering of SalarySystem would be based on the estimated amount of work to finish the project. And because ScanSys would be reluctant to sign the contract if the price was high, RussCo decided to make low estimates:

> It was a point to make low estimates in order to get the contract.
>
> — *Former RussCo project manager*

Through the SalarySystem project, RussCo reported the progress of the project to ScanSys on a weekly basis. In the estimation phase of the project, the reports from RussCo indicated that the project was going well, with no untoward problems. These reports were seen by ScanSys to be correct, and no cross verification was made by the ScanSys project staff:

> It was reported that there was no problem, [the ScanSys project manager] did not have any reason to check more thoroughly.
>
> — *ScanSys company manager*

Later, when the project almost failed, these reports were seen to have been incorrect and provided a misleading picture of the project progress.

3.4.3 The near-breakdown

Five months after project start the staff at ScanSys was not pleased with the project. The software deliveries from RussCo contained lots of bugs, screens were without all the requested functionality, and several important modules were missing. These factors made it impossible to test the system as a whole because the Norwegians did not have access to all the necessary modules. ScanSys contacted RussCo and asked critical questions, indicating that the project was delayed. The Russians did not agree though, but claimed that everything was on schedule. Then ScanSys was told that the project was three weeks late, and after two more weeks RussCo requested six more months to finish the project. That message made the Norwegians go to Russia, and the two companies had a hard confrontation. ScanSys blamed RussCo for the delay, and RussCo had no arguments against this. The RussCo project manager was then asked to come in, and he was blamed for the delay and removed as project manager in front of the Norwegians. The ScanSys staff was surprised by this episode:

> It was a hard situation. We are not used to it in Norway.
>
> — *ScanSys project manager*

Later the project manager quit RussCo. He did so because he was moved from the development department to the marketing department, and understood he had no future in the company. During the meeting, the Russians offered to finish the project without being paid for additional time:

> We added more people to the project, and ran it on the "company pocket".
>
> — *RussCo general manager*

One of the additional staff who was put full-time on the project, was the "Delphi Guru". He possessed both domain and technical knowledge necessary in order to finish the project.

3.4.4 The recovery phase

Prior to the near-breakdown of the project, the software deliveries from RussCo contained lots of bugs, but the Russians found it difficult to identify them because they did not understand how SalarySystem was used by the end users. Later, when ScanSys understood that RussCo lacked the appropriate understanding of SalarySystem even if they had all available written documentation about the system, they offered to make *test cases*. Each of the around 15 test cases describes a scenario or operation in SalarySystem considered difficult. The test cases are described in section 4.6 on page 78.

Another change in the project was the introduction of TestTool; an application which helps organizing the development of software projects. The use of TestTool had several advantages. One example is that it made the Norwegians more able to understand the Russians' problems, another that the status reporting from Russia improved. TestTool is described in section 4.5 on page 63.

Three months after the near-breakdown, three ScanSys staff; the project manager, the test coordinator, and a developer went to Russia. This meeting took place nine months after project start, and was very useful for the Russians:

> The Russians almost attacked me with lots of questions about how the product was supposed to be. Then I had to ensure them I was there to help them. After a while we could start to sort out the problems.
>
> — *ScanSys test coordinator*

During the meeting, the Norwegians explained details about SalarySystem and the Norwegian tax and salary rules which the Russians did not understand. In addition to the Russians' increased understanding of SalarySystem, the meeting also had other advantages:

> Before I went to Russia, all the different names were the same person to me. I could not tell them apart. I think we should have met before and had a party. When you know the people from the other site, you are not too "pushy". Instead of being in doubt whether they are able to finish the job, you think it is okay. They will surely fix it. You are stricter with people you don't know, and more relaxed with people you know.
>
> — *ScanSys test coordinator*

Through the meeting, the Russians came to understand more about the calculation engine being developed in Norway, the ScanSys test coordinator educated the testers, and the ScanSys project manager told them what parts of the project that was most important to finish. Both in Norway and Russia, the meeting was considered very positive. It was also generally agreed upon that the meeting took place too late in the project:

> If this kind of meeting had been held earlier in the project, it would have changed the project dramatically.
>
> — *The "Delphi Guru"*

After this meeting, more face-to-face meetings took place both in Norway and Russia. Also, ICQ[2] was installed in order to provide answers to short questions from the Russians. But even if the test cases, the introduction of TestTool, more frequent meetings, and ICQ improved the project performance, there were still problems in the project—generally related to lack of understanding of SalarySystem. 14 months after project start one of the project staff said:

[2]ICQ is an instant messenger. See `www.icq.com`.

> Even this week we have learned new features.
>
> — *RussCo project member*

At that time, the system still did not cover all functionality; some parts were still under implementation. The severity of the problems was little though.

3.4.5 Interdependency

When the project started, ScanSys were in a position of strength since they had several software companies to choose from while RussCo was the "weak part". The contract of the project, signed two months after project start, illustrates this. According to this contract, the project was divided into eight phases where the eighth phase was delivery of the pilot version. Each phase had a maximum number of man hours, and if this work amount was exceeded, RussCo had to pay for the rest of the phase. If the job was completed in less time, the unused man-hours could be carried over to the next phase. Also, both companies had the possibility to stop the project after any finished phase. 14 months after project start, when the amount of work had exceeded the estimates by more than six times, the project was still in phase seven. ScanSys then had the choice of stopping the project after the seventh phase, and thereby getting the SalarySystem software at low cost. However, ScanSys got more and more dependent on RussCo as the project moved on, and needed RussCo to develop the software:

> If the testing fails, I have no option.
>
> — *ScanSys project manager*

Because of knowledge sharing and RussCo's clever developers, ScanSys realized the value of the RussCo relationship. This understanding made ScanSys decide to finish phase eight internally, but still pay RussCo $14 000 according to the contract. That was possible because the phase was mainly concerned with testing; a job ScanSys could do in Norway. The ScanSys company manager said the following about this decision:

> The $14 000 is meant as a compensation for the already exceeded amount of work. What [ScanSys] gain is future co-operation.
>
> — *ScanSys company manager*

Because of ScanSys' dependency on RussCo, the ScanSys general management felt they had to act this way in order to ensure the success of possible future projects with RussCo.

By the end of the SalarySystem project, new GSO projects between ScanSys and RussCo were launched. As a result of the willingness to acknowledge and admit to the previous problems, and learning from these experiences, the project had become advantageous for both companies.

Chapter 4

Trust in the SalarySystem project

In this chapter, the attributes of trust are applied to the findings from the SalarySystem project. This analysis considers the project in a chronological way, based upon the structure from chapter 3. This is done to be able to look at trust as a process which develops over time.

4.1 Preconditions

When ScanSys decided to outsource the development of the new SalarySystem to Russia, the decision was made for two reasons: ScanSys needed the software to be developed at a low cost, and the company lacked Delphi developers inside its organization. Yet, other factors were considered too. Geographical distance in GSO relationships was one of the issues being discussed. However, since the difference in time zones between Norway and Russia is only two hours, this problem was not considered important by the ScanSys general management. The close temporal distance was actually one of the main reasons for the choice of Russia instead of more distant countries like India and China. Other problems associated with GSO relationships like issues of communication, knowledge sharing, project management, coordination, and travel costs were also taken into consideration by the ScanSys general management. Nevertheless, after having estimated a management overhead at about 30%, they still found that GSO was cost saving.

Although the general strategy for growth at ScanSys was through acquisitions of existing companies, this strategy was seen to be too risky because of the substantial investments required. Instead they found "direct outsourcing to supplier firms" to be a more appropriate strategy because less investments had to be made.

Even if Russia had the fifth highest software piracy rate worldwide in 2001 (Business Software Association 2002), ScanSys did not worry about that. They found SalarySystem only to be useful inside Norway because of the embedded Norwegian tax and salary rules:

> The old source code is badly written and has no value outside Norway because of the tax rules.
>
> — *ScanSys developer*

Thus, the risk of piracy was considered irrelevant.

4.1.1 Trust in GSO as a phenomenon

In accordance with the above description, the ScanSys general management had a positive attitude towards GSO during this time. This thesis interprets that attitude to be calculation-based trust since ScanSys found the associated risks to be suppressed by the opportunities of GSO. Thus, a GSO project was considered an opportunity. However, since the ScanSys–RussCo relationship was not yet established, there was no one to trust—neither individuals nor organizations. Thus, the trust at that time is seen to be in GSO as an advantageous phenomenon. Furthermore, since the SalarySystem project later ran into problems and was delayed by more than a year—partly because of issues not considered important by ScanSys—this thesis finds the 30% management overhead to be too optimistic on behalf of GSO projects. Consequently, the estimated risk that ScanSys associated with GSO deviated from the objective risk.

Although ScanSys had limited experience in outsourcing, their faith in GSO was relatively strong. However, since knowledge-based trust can stem from other sources than first-hand experience, it is reasonably to believe that their knowledge came from other channels. In turn, this knowledge was important for the Norwegians' feeling of predictability about the future GSO project. Later, when the Norwegians visited RussCo for the first time, the positive expectations they had about GSO as a phenomenon resulted in a positive attitude towards RussCo.

4.2 The choice of RussCo

After Russia was chosen as the country to outsource to, some people from the ScanSys general management went to Russia to look for their future business partner. Their most important criteria were the company had to be of similar size as ScanSys and having a Western organization style. The latter included the company being not too hierarchical. RussCo fulfilled these criteria to a large extent. In general, the Norwegians may be seen to look for a company similar to ScanSys. In this thesis, this wish is interpreted as a way of predicting the outcome of the project in a better way. According to the attribute of trust called knowledge, knowledge-based trust can be built as a result of first-hand experience, reputation, surface attributes, and stereotypes. When an individual recognizes characteristics of another which he or she is familiar with, a feeling of predictability may appear more likely compared to situations where everything is new. For instance, after visiting Moscow when looking for a company, the ScanSys staff found the city to be larger and more crowded compared to Norwegian cities. And even if St. Petersburg to some extent shares these characteristics, the Norwegians found the city to be more similar with their prior experiences:

> Compared to Moscow, I felt more at home in St. Petersburg.
>
> — *ScanSys company manager*

Thus, feeling "at home" was important in order to predict the new situation. So, when the Norwegians visited RussCo and found characteristics they could relate to in a positive way, this thesis interprets this gut feeling as knowledge-based trust. Though, other attributes of trust were important too.

Believing that recognizable characteristics are beneficial also has to do with goodwill-trust. This attribute of trust identifies the general tendency to trust others to be important for how likely an individual will feel good about others. Thus, when the Norwegians felt good because RussCo was seen to have a Western organization style, this thesis suggests the Norwegians' general tendency to trust others was important for this feeling to appear. Furthermore, during this early phase of the relationship, ScanSys was the strong part meaning they were operating in a buyer's market. They could choose from a number of software supplier companies and demand what the pilot contract should look like. Because of this strong position, it might have been easier to trust RussCo (See section 2.2.5 on page 25).

The pilot project that preceded the SalarySystem project, were important in order to build competence-based trust. However, the importance of competence-based trust is discussed in more detail in section 4.3 on page 49.

4.2.1 The "Cultural Liaison"

The "Cultural Liaison" was important when the ScanSys general management had to choose among the different software suppliers in Russia.

His proximity to Russia regarding knowledge of language and culture, enabled him to help the Norwegians decide what companies would be able to perform the SalarySystem project. For instance, he knew some of companies which had an inadequate information infrastructure or were notorious sellers of pirate copied software. Furthermore, his proximity to Russia made him more capable to interact and learn about the companies they visited compared to the Norwegians, even if they were all in the same room. For instance, some of the Russians spoke English poorly, so he could use his Russian language to improve the communication. In turn, he could pass on his impressions to the Norwegians. As a result, his presence was important for the Norwegians' degree of competence-based trust. Thus, the assumption that the ScanSys general management trusted the "Cultural Liaison" is implicit in this interpretation.

Another important role of the "Cultural Liaison", was helping the Norwegians act appropriate in Russia. Because of differences between the two countries with respect to ways of doing business, he was useful for teaching them how to deal with these differences. For instance, drinking alcohol at business meetings is accepted in Russia which is not the case in Norway. Also, Russia has a black economy existing in parallel with the legal part. These differences were difficult to handle for the inexperienced Norwegians, so the "Cultural Liaison" was helpful in telling them how to behave. Since the Norwegians to a larger extent behaved as anticipated, it was easier for the Russians to build knowledge-based trust in them. Thus, the "Cultural Liaison" was helpful in building trust both in Norway and Russia.

4.2.2 Trust in Russia

During the early phases of the project—at the time ScanSys visited RussCo for the first time—this thesis suggests that RussCo got a strong calculation-based trust in ScanSys as a future customer. This interpretation stems from several circumstances. Prior to the SalarySystem project, RussCo had experienced problems in getting new customers. When RussCo in 1995 joined a software seminar in Finland in order to establish contacts with potential new customers, the RussCo general manager described this effort as follows:

> After my presentation, many participants wanted to give me projects, but that was only words.
>
> — *RussCo general manager*

More specific, RussCo wanted to get into the Scandinavian market because of mixed experiences with some of their customers from the US:

> It is hard to work with the Americans. They move fast and want everything now.
>
> — *RussCo general manager*

In contrast, Scandinavians were seen to prefer long-term relationships. Also, the close cultural and geographical distance was considered beneficial:

> It is nice to work with Scandinavians because we both speak the same broken English, and not as fast as the Americans. Finland, Russia, and Norway are more alike.
>
> — *RussCo general manager*

Furthermore, because of this strong wish to engage in the project with ScanSys, other kinds of trust seem to have been suppressed at RussCo. Compared to the ScanSys staff, whose trust came from a number of different sources like learning to know their future business partner better and launching pilot projects, the RussCo general management felt good about the project solely because of the calculation-based trust. This strong degree of trust was also present during the near-breakdown phase. When ScanSys blamed RussCo for the slow progress in the project, the RussCo general management agreed to finish the project without being paid for additional time. However, the interaction that had taken place since the beginning of the SalarySystem project suggests that attributes like predictability and knowledge were also present to some extent.

4.3 The first meeting

The first meeting in Norway took place two months before the SalarySystem project started. During the meeting, several aspects related to trust came into play.

4.3.1 Trust in Norway

From the Norwegian point of view, the purpose of the first meeting was to transfer knowledge about SalarySystem and its domain from Norway to Russia as well as learning more about RussCo's competence as a software supplier. Using the attributes of trust to explain these purposes, the meeting would be useful to build competence-based trust in the Russians. Furthermore, the meeting was also supposed to build knowledge-based trust since the intention was to learn more about the Russians.

As the meeting went by, the Norwegians were impressed by the Russians' skills—especially those of the "Delphi Guru". Also, when the Norwegians explained details about SalarySystem, they felt that the Russians better understood what the project was about. They were convinced that the Russians would be able to do the job, so competence-based trust was built. When it comes to knowledge-based trust, the description of the knowledge attribute suggests that interaction itself is a source of trust. Although the results from the case study not specifically address the advantages from the first meeting, one of the ScanSys staff described the problem of lack of knowledge in a more general way:

> A problem related to distance is that before you meet the other team, you cannot imagine their faces.
>
> — *ScanSys test coordinator*

Based on this statement, this thesis suggests that knowledge-based trust was built during the first meeting because of the interaction that took place.

Considering goodwill trust as a reason for this high degree of trust, the Norwegians in the SalarySystem project may be seen to have had a general tendency to trust others. On the other hand, because ScanSys was the strong part in the relationship during this phase of the project, it might have made them more likely to trust the Russians' competence as mentioned earlier. Another reason might have been that the Norwegians themselves did not realize what skills were needed to perform the SalarySystem project. By the end of the project, the ScanSys company manager said:

> Looking back, the SalarySystem project was much more complicated than we knew. We knew what we wanted, but did not realize the complexity.
>
> — *ScanSys company manager*

Since the project later ran into problems while the Norwegians did not understand why, their predictability about the Russians is seen to have been low in the beginning of the project. Still, the Norwegians' degree of trust was high. Thus, there was a difference between their degree of trust and the Russians' ability to perform the SalarySystem project, so trust and predictability are not the same. This situation where the Russians' ability to perform the project deviated from the Norwegians' predictions is illustrated in figure 4.1 on the facing page. It is important to notice that the figure only considers the predictability attribute of trust.

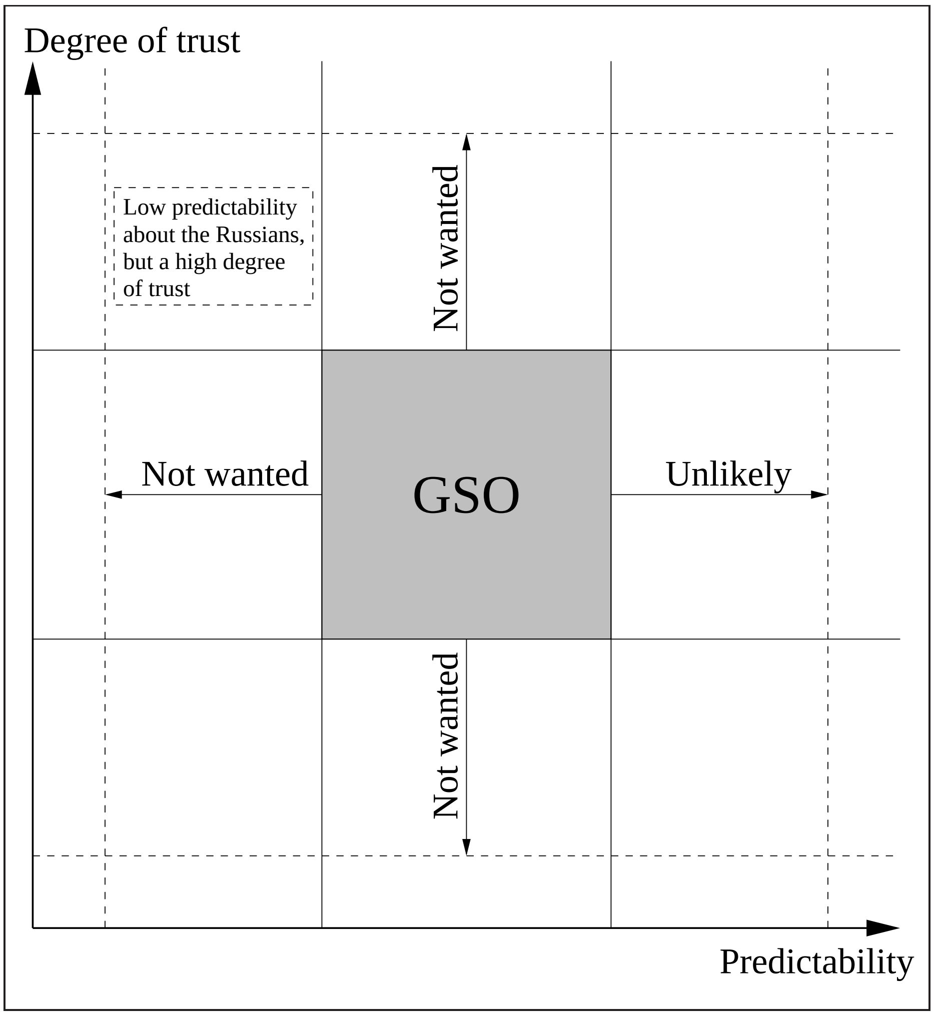


Figure 4.1: The Norwegians' trust was high compared to their predictability.

A possible betrayal

After the return to Russia, the project manager and the "Delphi Guru" spent most of their time on other projects, so their knowledge about SalarySystem was not adequately shared with the rest of the Russian project members as assumed by the Norwegians. Thus, the Norwegians' expectations about the Russians were violated. According to the definition of betrayal, the violation of the Norwegians' well-being also had to be voluntary. However, considering the strong Russian calculation-based trust during this phase of the project, betraying the Norwegians would not be beneficial. Instead, this thesis identifies inadequate internal communication at RussCo as a reason that might explain the reassignment of the two RussCo staff. Since the Russians lacked understanding of the SalarySystem project, they failed to see the complexity too. Later, when the project manager realized the project was running into problems, he requested more people from the RussCo general management. Unfortunately, he did not succeed in making them understand the seriousness of the problems:

> I asked [the RussCo general manager] to give me more people, but I only got [RussCo developer IV]. My mistake was that I was not strong enough.
>
> — *Former RussCo project manager*

This inadequate internal communication at RussCo was also confirmed by the general manager:

> [Five months after project start] we picked up the project that was near a failure.
>
> — *RussCo general manager*

When the RussCo general management during the near-breakdown phase realized the project needed more manpower, more people—including the "Delphi Guru"—was assigned to the project. As a result of the above interpretation, the RussCo staff cannot be said to have voluntarily intended to betray the Norwegians by allocating too few people to the SalarySystem project. Instead, the poor internal communication at RussCo is seen to have been the reason.

4.3.2 Trust in Russia

During the early phases of the project, the Russians had a strong calculation-based trust in the Norwegians because they wanted the contract of the SalarySystem project. As a result, they wanted ScanSys to look upon RussCo as a competent software supplier. In this thesis, the role of the "Delphi Guru" is interpreted to be a way of demonstrating such competence. During the meeting, his job was to learn about SalarySystem, and—by suggesting how to develop the new system—demonstrate RussCo's competence as a software supplier. Furthermore, after the meeting in Norway, he was reassigned to other projects. Hence, this interpretation is based upon the assumption that it would have been better for the project if his recently obtained knowledge about SalarySystem had been available for the other Russian project members. This view was later confirmed by the "Delphi Guru" himself when discussing the evaluation of the estimates of the SalarySystem project:

I did not see the estimates when they were finished. If I had seen them, I might have been able to identify the problem.

— *The "Delphi Guru"*

The two RussCo staff who joined the meeting in Norway, had SalarySystem and its domain knowledge explained by the Norwegians. During the meeting, the Russians increased their knowledge about the project, but the information they brought back was not very extensive. Furthermore, the Russians asked questions the Norwegians could not answer later on in the project. As a result, the Russians started to doubt the Norwegians' knowledge about the project. The assumption that the poor Norwegian domain knowledge could complicate the SalarySystem project was not considered when RussCo after the return to Russia made the estimates. Thus, the Russians' strong calculation-based trust might have made the low degree of competence-based trust less important.

4.4 The estimation process

Several aspects related to trust came into play during the estimation process. In this thesis, issues concerning the Software Requirements Specification, or SRS, are seen to illustrate some of these aspects. At this stage of the SalarySystem project, the contract was not signed yet, and the Russians lacked the appropriate understanding of the product. For these reasons, the Russians wanted to show their competence to increase their chance of having the SalarySystem contract signed as well as future contracts, and they made the SRS. The Russians made the SRS by looking at the running version of SalarySystem and analyzing the source code. When the Norwegians got the document, they were impressed by its size and level of technical detail. But when they later—during the near-breakdown phase—found that the Russians lacked the appropriate understanding they were believed to have according to the SRS, the degree of trust in the Russians decreased. The SRS is described below.

4.4.1 Contents of the SRS

The SRS consists of three main parts in addition to the introduction: a workflow description, a section describing the reports, and at last a database description. These three parts will all be described in detail below in the sequence they appear in the SRS.

The structure of the SRS, and especially the introduction, is partly based on the IEEE standard 830-1998[1] (IEEE Std 830-1998 1998). This

[1]The title of the standard is "IEEE recommended practice for software requirements specifications".

standard recommends what an SRS should look like regarding contents and issues to be addressed. However, apart from the chapter headings, the SRS does not follow the standard to a large extent. IEEE Std 830-1998 (1998, p. 3) identifies five issues an SRS should address: *functionality*, *external interfaces*, *performance*, *attributes*, and *design constraints imposed on an implementation*. Regarding *functionality*, the GUI of the old SalarySystem is specified using several flow diagrams. Also, the reporting function of the system is specified with respect to what fields to report from, and how to filter the data. When it comes to the next three issues; *external interfaces*, *performance*, and *attributes*, none of them are addressed. One feature of SalarySystem is the ability to export several reports and templates into Microsoft Word or Excel, but this feature is not mentioned in the SRS. The *design constraints imposed on an implementation* are addressed because both the choice of programming language and database solution are specified. The connections between the IEEE standard and the SRS are summarized in table 4.1.

Issue to be addressed	**Addressed in the SRS**
Functionality	• The GUI is specified using flow diagrams • The report function is specified
External interfaces	• Not addressed
Performance	• Not addressed
Attributes	• Not addressed
Design constraints imposed on an implementation	• Programming language and database solution are specified

Table 4.1: Basic issues recommended for an SRS, and how these issues are addressed in the SRS of the SalarySystem project.

Workflow description

The first part of the SRS considers the workflow of the system which is to a large extent described using flow diagrams. Every step a user has to perform is described within a box, and the boxes are connected with arrows. Also, some of these boxes have additional information attached to them in a tag. Such additional information may concern details about the workflow not directly connected to the use of the system, but rather the function of the system on a lower level, for example details of a certain function in the source code. The diagrams go through a GUI, or a group of connected GUIs, and describe how to perform a certain task from a user's perspective. That is, one diagram shows one function which may consist of several GUIs. The descriptions contain details about how to navigate through the GUIs, what buttons to press, where to fill in values, and what boxes to check. The diagram does also, to some extent, show which alternatives are possible for a given operation. Figure 4.2 on the following page shows a diagram from the SRS about how the payroll calculation is being performed. Figure 4.3 on page 57 is from SalarySystem, and shows the main GUI of the payroll calculation described in figure 4.2. The GUI contains some boxes and buttons, each providing a specific functionality for the user. The workflow diagram describes the functionality of all the GUIs related to payroll calculation.

The workflow diagram in figure 4.2 is the only information in the SRS about how to perform a payroll calculation, but since this functionality consists of other features too, the description of the payroll calculation may be considered incomplete. For example, if the value of the field "Skatt"[2] is set to another value than "Normal Skatt"[3], another functionality, and another workflow, would appear. Because of the Norwegian tax and salary rules, such new GUIs appear in SalarySystem only when specific values and functionalities are chosen. This implies that the user has to be quite skilled to be able to access these GUIs. And because the Russians did not possess this knowledge, they found it hard and time consuming to find all the "hidden" GUIs. So even if the SRS was correct considering the functionality actually described, it failed to describe all the functionality in SalarySystem. Also, the workflow description was incomplete concerning what levels of SalarySystem it concerned, meaning every step from the database up to the GUI. Because only the GUI is interpreted, a thorough description of what happens at the database level, at the function level et cetera is missing. In the third part of the SRS the database tables are described. However, the two parts describing the workflow and the database are not connected in the SRS, and so there are no connections between these levels either.

Reports

SalarySystem has around 150 templates for different reports and statistics to be generated from its database. Examples of some standard reports include addresses of the employees, salary types, absence from work, and lists of creditors. There also exists a report generator where the user can design his or her own reports, but this module is not a part of the standard SalarySystem. It has to be bought in addition to the standard version. However, this feature was a part of the redesign, but it is not

[2]English: Tax.
[3]English: Normal Tax.

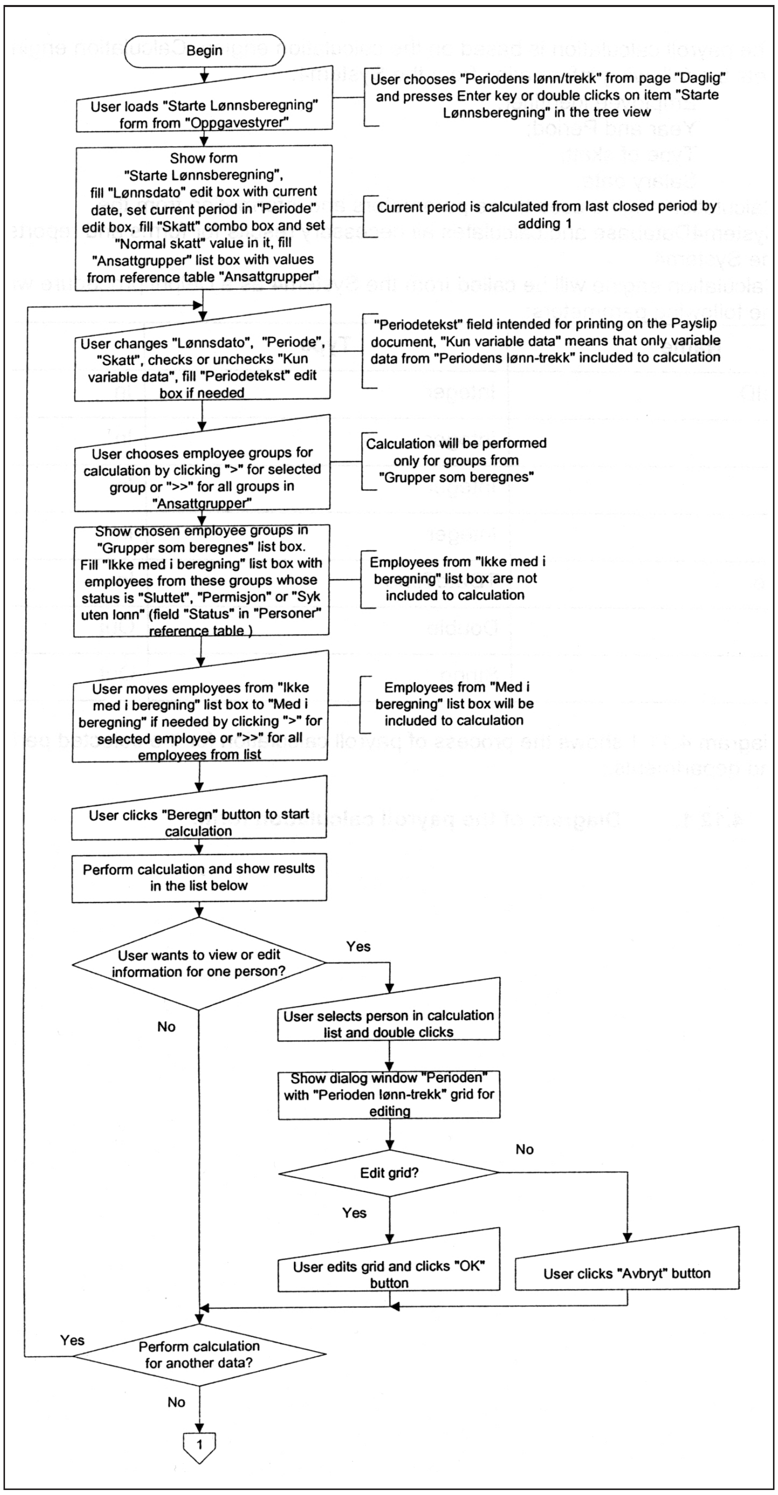
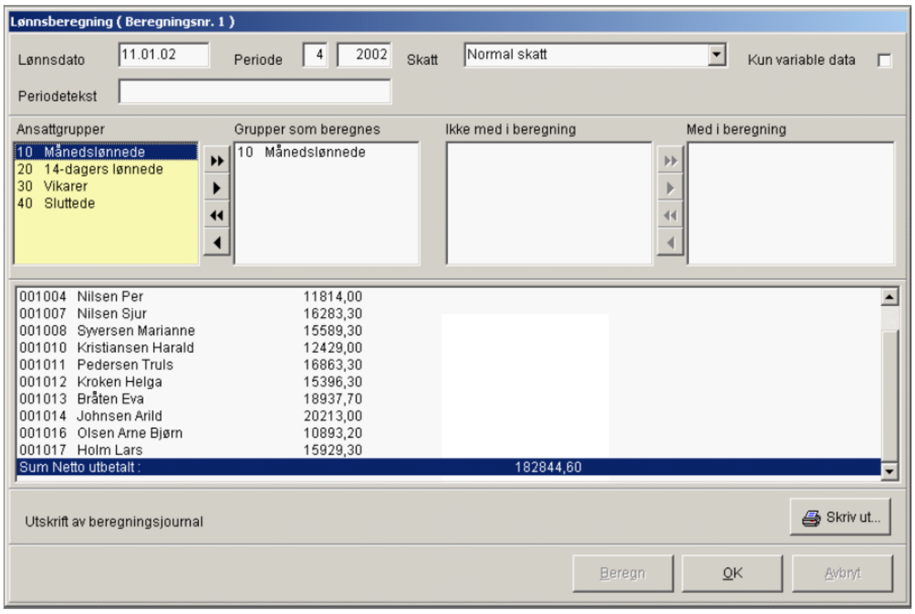

Begin
User loads "Starte Lønnsberegning" form from "Oppgavestyrer"
User chooses "Periodens lønn/trekk" from page "Daglig" and presses Enter key or double clicks on item "Starte Lønnsberegning" in the tree view
Show form "Starte Lønnsberegning", fill "Lønnsdato" edit box with current date, set current period in "Periode" edit box, fill "Skatt" combo box and set "Normal skatt" value in it, fill "Ansattgrupper" list box with values from reference table "Ansattgrupper"
Current period is calculated from last closed period by adding 1
User changes "Lønnsdato", "Periode", "Skatt", checks or unchecks "Kun variable data", fill "Periodetekst" edit box if needed
"Periodetekst" field intended for printing on the Payslip document, "Kun variable data" means that only variable data from "Periodens lønn-trekk" included to calculation
User chooses employee groups for calculation by clicking ">" for selected group or ">>" for all groups in "Ansattgrupper"
Calculation will be performed only for groups from "Grupper som beregnes"
Show chosen employee groups in "Grupper som beregnes" list box. Fill "Ikke med i beregning" list box with employees from these groups whose status is "Sluttet", "Permisjon" or "Syk uten lonn" (field "Status" in "Personer" reference table)
Employees from "Ikke med i beregning" list box are not included to calculation
User moves employees from "Ikke med i beregning" list box to "Med i beregning" if needed by clicking ">" for selected employee or ">>" for all employees from list
Employees from "Med i beregning" list box will be included to calculation
User clicks "Beregn" button to start calculation
Perform calculation and show results in the list below
User wants to view or edit information for one person?
Yes
No
User selects person in calculation list and double clicks
Show dialog window "Perioden" with "Perioden lønn-trekk" grid for editing
Edit grid?
No
Yes
User edits grid and clicks "OK" button
User clicks "Avbryt" button
Yes
Perform calculation for another data?
No
1

Figure 4.2: The workflow of the payroll calculation form.



Lønnsberegning (Beregningsnr. 1)

Lønnsdato 11.01.02 Periode 4 2002 Skatt Normal skatt Kun variable data

Periodetekst

Ansattgrupper	Grupper som beregnes	Ikke med i beregning	Med i beregning
10 Månedslønnede	10 Månedslønnede		
20 14-dagers lønnede			
30 Vikarer			
40 Sluttede			

```
001004  Nilsen Per               11814,00
001007  Nilsen Sjur              16283,30
001008  Sywersen Marianne        15589,30
001010  Kristiansen Harald       12429,00
001011  Pedersen Truls           16863,30
001012  Kroken Helga             15396,30
001013  Bråten Eva               18937,70
001014  Johnsen Arild            20213,00
001016  Olsen Arne Bjørn         10893,20
001017  Holm Lars                15929,30
Sum Netto utbetalt :                          182844,60
```

Utskrift av beregningsjournal

Figure 4.3: The payroll calculation form from SalarySystem—first part.

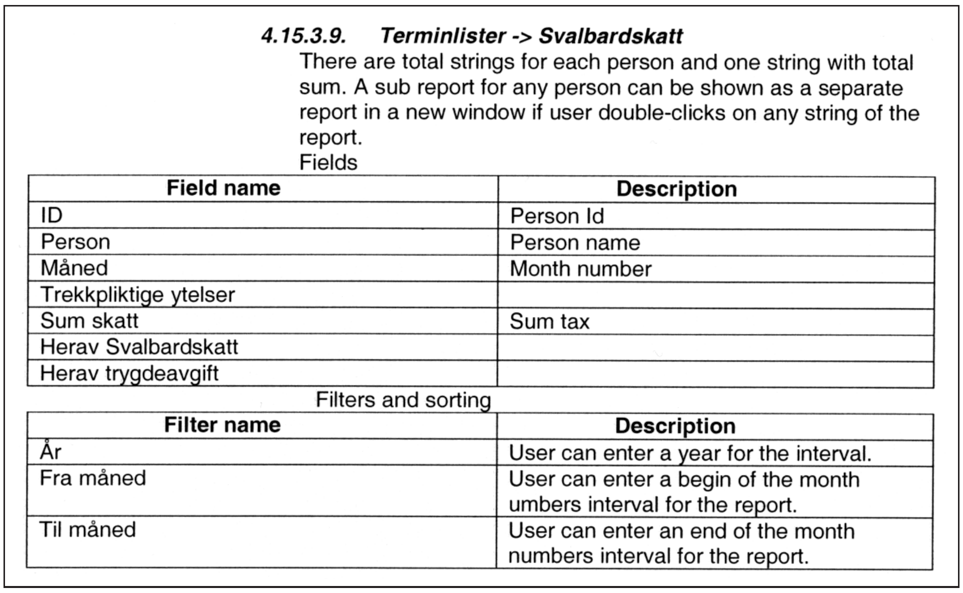
described in detail in the SRS. One reason for this, is that it is a relatively stand-alone module. And because the project was delayed, this feature was postponed to the end of the project and consequently not considered in detail in the SRS.

The second part of the SRS describes the standard reports. These are interpreted by a written description and a table showing the fields of the report in addition to the filters. The filters determine what values to be queried from the database. Also, the filters are used for specifying how to sort the data in the finished report. Figure 4.4 on the following page is an example of such a report description. This particular report concerns Svalbardskatt[4] which means tax being paid by persons living at Svalbard. As figure 4.4 shows, the fields in the report are given as well as their English translation. In other report descriptions in the SRS, the "Description" field is used for mentioning the valid values of the field. In the second part of the table, the query alternatives are described.

The reports were described by looking at the interface of SalarySystem, and they cover the functionality to a large extent. However, only 2/3rd of the 150 reports are described in the SRS. So even if the described reports are correct, the SRS is incomplete considering the reports.

Although it was important to describe the GUI in order to develop

[4]English: Svalbard tax. According to the Norwegian tax and salary rules, the tax in Svalbard is different from the tax in other regions of Norway.



4.15.3.9. Terminlister -> Svalbardskatt

There are total strings for each person and one string with total sum. A sub report for any person can be shown as a separate report in a new window if user double-clicks on any string of the report.

Fields

Field name	Description
ID	Person Id
Person	Person name
Måned	Month number
Trekkpliktige ytelser	
Sum skatt	Sum tax
Herav Svalbardskatt	
Herav trygdeavgift	

Filters and sorting

Filter name	Description
År	User can enter a year for the interval.
Fra måned	User can enter a begin of the month umbers interval for the report.
Til måned	User can enter an end of the month numbers interval for the report.

Figure 4.4: The description of the report for Svalbardskatt.

the correct reports, the SRS lacks descriptions of what happens at other levels of the system. Because the data in the reports are collected from different tables of the database, the Russians had problems finding the data. A Russian developer responsible for the reports said:

> One challenge was to know where to get the data from.
>
> — *RussCo developer II*

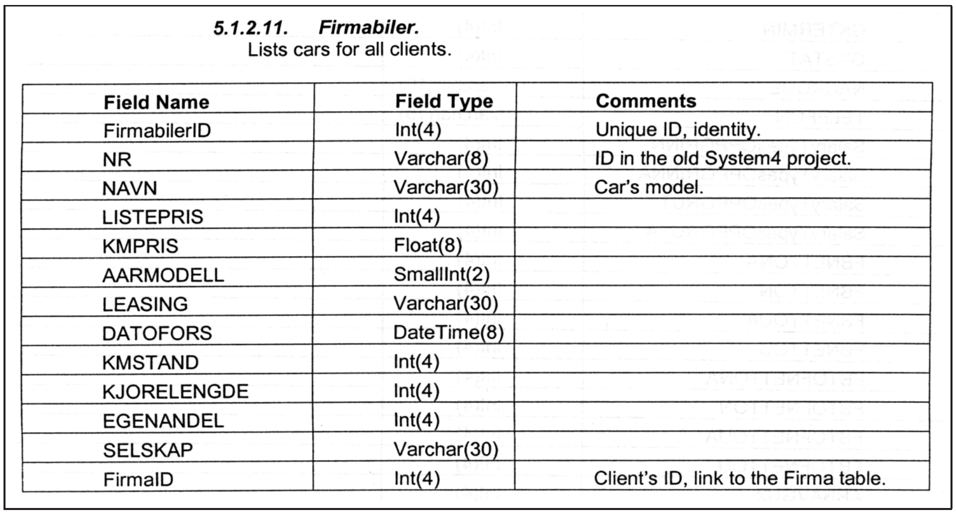
Another reason for this problem was the fact that all database fields, tables et cetera in the old source code were written in Norwegian. In the initial phase of the project, ScanSys and RussCo decided to continue using Norwegian in the source code of the redesign product too. This was done to maintain consistency between the old and the new system. Because these fields and tables are usually recognizable from their name, it hampered the Russians' understanding of the system. One of the developers found the language issue crucial, so she bought herself a book called something like "Norwegian for Beginners" to learn Norwegian. This way, she hoped to address the language problem. The language issue is also reflected in the SRS where the field names are translated into English in the "Description" field (see figure 4.4). Also, the business logic encapsulated into the source code made the Russians struggle even harder to understand the working of the system:

There are about 150 reports in SalarySystem, and 50% of them are difficult to understand.

— *RussCo developer II*

Database tables

The last part of the SRS document is a description of the SalarySystem database. The description consists of 72 tables similar to figure 4.5. First there is a short written description about the purpose of the table. Then the database table itself is described by its field names and the corresponding types and comments. The "Comments" field is used for describing which field in the table is the unique identifier of the table, what values are allowed in the field, and what function the field has in the system. One example of the latter is "Car's model".



5.1.2.11. Firmabiler.
Lists cars for all clients.

Field Name	Field Type	Comments
FirmabilerID	Int(4)	Unique ID, identity.
NR	Varchar(8)	ID in the old System4 project.
NAVN	Varchar(30)	Car's model.
LISTEPRIS	Int(4)	
KMPRIS	Float(8)	
AARMODELL	SmallInt(2)	
LEASING	Varchar(30)	
DATOFORS	DateTime(8)	
KMSTAND	Int(4)	
KJORELENGDE	Int(4)	
EGENANDEL	Int(4)	
SELSKAP	Varchar(30)	
FirmaID	Int(4)	Client's ID, link to the Firma table.

Figure 4.5: The SalarySystem database table for firmabiler (English: company cars).

The database tables are described one by one, leaving out the connections between them. However, one exception is that some tables have a text like "Client's ID, link to the Firma table." connected to them. These kinds of descriptions are limited to only a few tables though. Also, all key attributes in the tables have the comment "Unique ID, identity", so the database schema would be possible to draw using the information already in place. However, the fact that the connections between the different tables in the database was not explicitly described, may have increased the difficulties of knowing where to get the data for the reports from mentioned in section 4.4.1 on page 55. Adding an ER diagram (Elmasri and Navathe 2000) for the SalarySystem database would probably have been found useful for the developers implementing the reports.

4.4.2 The role of the SRS

The SRS document was made as a result of the RussCo analysis of SalarySystem, and the first version was delivered to the Norwegians one month after project start. The SRS itself defines its purpose to be a "specification for the SalarySystem project", and thus provide a framework within which the development process could continue. However, this thesis interprets that the primary purpose of the SRS was to establish to ScanSys their technical competence and capability in software development and gain the contract, rather than as a document that could serve as a basis for estimation and to guide the future development work. Both the SRS and the RussCo estimates were made before the contract of the project was signed, and since RussCo had tried to get into the Scandinavian market for some time, this contract—and the relationship with ScanSys—was seen to be important.

Calculation-based trust has to do with the assessment of whether the trustee is able to perform an action that is beneficial (Rousseau, Sitkin, Burt, and Camerer 1998; Kim and Prabhakar 2000; Sabherwal 1999). This description fits with how the RussCo staff considered the relationship with ScanSys. However, while getting the contract was a matter connected to the organizational level of the project, the calculation-based trust was also important for the testers and developers on the "micro level". That is, the organizational level below the RussCo general management. Because RussCo wanted to show its competence, and both the SRS and the estimates were made before the contract was signed, these two documents were influenced by the wish to get the contract.

By applying the attributes of trust, the intention of making the SRS can be seen as an attempt to make the Norwegians build competence-based trust in the Russians. One argument for this interpretation is the extent of which the Russians found the document useful during implementation:

> The SRS was only useful for common questions, for example what report to create and its function.
>
> — *RussCo developer III*

Another argument is the role of the SRS with respect to software methodology:

> [. . .] development happened at the same time the SRS was made.
>
> — *RussCo developer IV*

Because the analysis and design phases of software development usually precede the development phase (Sommerville 2001), it is unreasonable to believe that the purpose of the SRS was to support the development. Also, one of the developers was explicitly told by his supervisor not to use the SRS in his work. He summarized the role of the SRS during the development process in the following way:

> Forget about the SRS, just fix the bug.
>
> — *RussCo developer III*

Still, the Russians spent much time making the SRS as well as revising it according to the Norwegians' evaluation. Furthermore, the attempt to base the SRS on IEEE Std 830-1998 (1998) is interpreted to be a way for RussCo of demonstrating competence-based trust too. Because IEEE is a well-known authority within software development, using their standard is a way of being associated with its competence. However, since RussCo did not follow the standard to a large extent when the SRS was written, this is found to be an attempt of demonstrating competence they did not possess. However, ScanSys later adopted the SRS template internally because they found it to be professional.

The evaluation of the SRS at ScanSys took place in the following way: The Norwegian project members looked through the document page by page and checked whether it was correct. The errors they found was put into a report and sent to RussCo. They had much iteration before the SRS was seen to be satisfying. During a period of three months, the SRS was updated ten times. However, the Norwegians did not check either for missing parts or considered if the detail level was appropriate for RussCo to be able to develop the system. The ScanSys staff was impressed by the quality of the document, and convinced that RussCo was technically capable of doing the project. One of the Norwegians who joined the evaluation said:

> It was a very accurate document, and I was impressed that [RussCo] managed to make it with the limited amount of information they had available. The quality of the SRS proves that [RussCo] are professional.
>
> — *ScanSys test coordinator*

According to this quote, RussCo succeeded in building competence-based trust although the focus on data flow and technical issues later turned out to reflect the too narrow focus of the RussCo staff because other important issues were ignored. Given this incompleteness of the SRS, the evaluation of the document in Norway did not reveal these aspects. Instead it made the Norwegians confident about the Russians' competence.

Goodwill is seen to be important for an individual's tendency to trust others. Dependent on the personalities of the ScanSys staff evaluating the SRS, they could assume (1) the Russians possessed knowledge about the rest of the system too, it just was not included in the SRS, or (2) the SRS was incomplete, so they did not possess the appropriate knowledge of the system. These two examples thus illustrate two opposite points of view dependent on an individual's tendency to trust. However, even if the ScanSys staff assumed (1) rather than (2), the conclusion that they had goodwill trust does not immediately make sense. Because the evaluation of the SRS was influenced by a number of other factors like lack of domain knowledge and commitment to the project, such a point of view would be to jump to conclusions. Furthermore, since powerful organizations are more willing to trust then weak ones (Sydow 1998), and ScanSys had the upper hand during this time of the project, these reasons might just as well explain the benevolence of the ScanSys staff. Thus, goodwill trust is difficult to identify based upon the above description. Moreover, (1) is similar to how predictability-based trust emerges: a way of predicting behavior is by observing patterns of behavior, and expecting them to continue (Maguire, Phillips, and Hardy 2001). Since the different parts of the SRS were incomplete rather than wrong, it may have made the Norwegians expect that the Russians' competence were present. Thus, the Norwegians had predictability-based trust in the Russians too.

The SRS and the estimates made the Norwegians more able to understand the Russians' knowledge of SalarySystem and its domain. Since these documents were detailed with respect to several aspects of the Russians' knowledge, they can be looked upon as structural mechanisms. The two documents gave the Norwegians a chance to review the Russians' understanding of the old system as a result of the evaluation process. Also, the weekly reports from Russia are considered structural mechanisms. However, considering the near-breakdown phase and the Norwegians not knowing about the slow progress of the project, this thesis interprets these structural mechanisms to be insufficient during the estimation process because of the low degree of actual predictability they provided.

Since the SRS creation took place in the initial phase of the project, establishing competence-based trust was important for RussCo. But as time went by and the software deliveries from Russia contained lots of bugs, the competence-based trust disappeared. Betrayal is defined as "a voluntary violation of mutually known pivotal expectations of the trustor by the trusted party (trustee), which has the potential to threaten the well-being of the trustor" (Elangovan and Shapiro 1998, p. 548). According to this definition, betrayal takes place only when the expectations of the trustor are violated *and* the action is voluntary. In the project, the Russians were found not to possess the knowledge about SalarySystem expected by the Norwegians, and so their degree of competence-based trust decreased. However, the Russians did not necessarily *intend* to lie about their competence. The interpretation of this thesis is that the Russians initially in the project did not have the knowledge about SalarySystem which they tried to show in the SRS, but that they believed they were going to learn before the project suffered. However, they underestimated the complexity and the amount of business logic of SalarySystem:

> We missed some forms in the beginning of the project. We found almost all of them, and their calculations were quite right, but we underestimated the complexity of some forms. Some examples of complex forms is the one about absence from work, and about complex custom control. These are defined by data from the database. At first glance they look easy, but we had to implement three to four variants before we made it work correctly.
>
> — *RussCo project coordinator*

Thus, considering the deviation between the SRS and the RussCo competence an act of betrayal, would not fit with the definition. However, the degree of competence-based trust the Norwegians had in the Russians decreased.

4.5 The near-breakdown

The use of TestTool is seen to illustrate some of the issues of trust during the near-breakdown. Below, TestTool is described in detail with an emphasis on these issues. The description is followed by an analysis of trust during the near-breakdown.

4.5.1 Properties of TestTool

TestTool is an off-the-shelf product[5] which helps software projects organizing the development process. It can be used to manage bugs and feature requests as well as other issues throughout the development cycle. A development project using TestTool has a database that can be accessed by all assigned project members. TestTool also has a web-based client, which makes it suitable to support virtual teams since the database can be accessed from wherever there is an Internet connection. In Norway all project members had access to the web-based version of TestTool, while all the Russian project members had access in addition to the company management. The TestTool database was located and administrated at RussCo. RussCo also initiated the use of the application in the project.

Figure 4.6 provides an overview of the use of TestTool. As the figure shows, the work taking place in a project concerning development and testing is registered. In turn, TestTool uses these data to produce output which is useful for the project members.

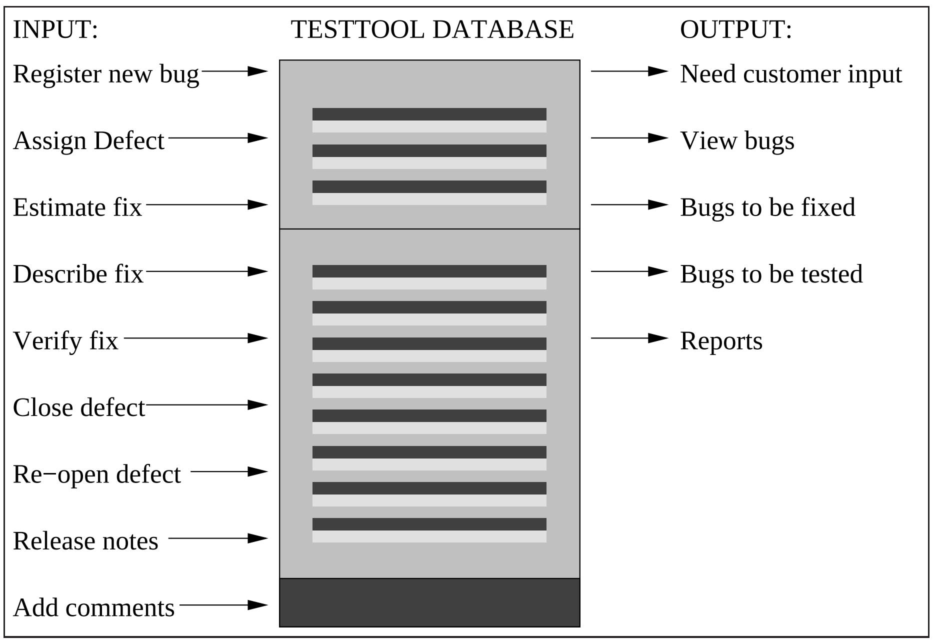


Figure 4.6: Input and output in TestTool.

The use of TestTool in the SalarySystem project had several advantages: it enabled the Norwegians to overlook the work taking place at the other site as well as the development process itself, it increased the

[5]Off-the-shelf software products are made for a large number of users in contrast to software made for a special purpose.

understanding of SalarySystem at the Russian site, and improved the reporting of project status. Most of the project members were satisfied with the use of TestTool in the project:

> [TestTool] makes it easy to assign bugs, cooperate, easy to see how many tasks you must solve, how many bugs et cetera.
>
> — *RussCo developer II*

Bugs in TestTool

This section describes how bugs are represented in TestTool. The description is divided into two parts; one about the data fields of the bugs, and one about the actions that can be performed on them. TestTool allows customization to a large extent, which has taken place in the project:

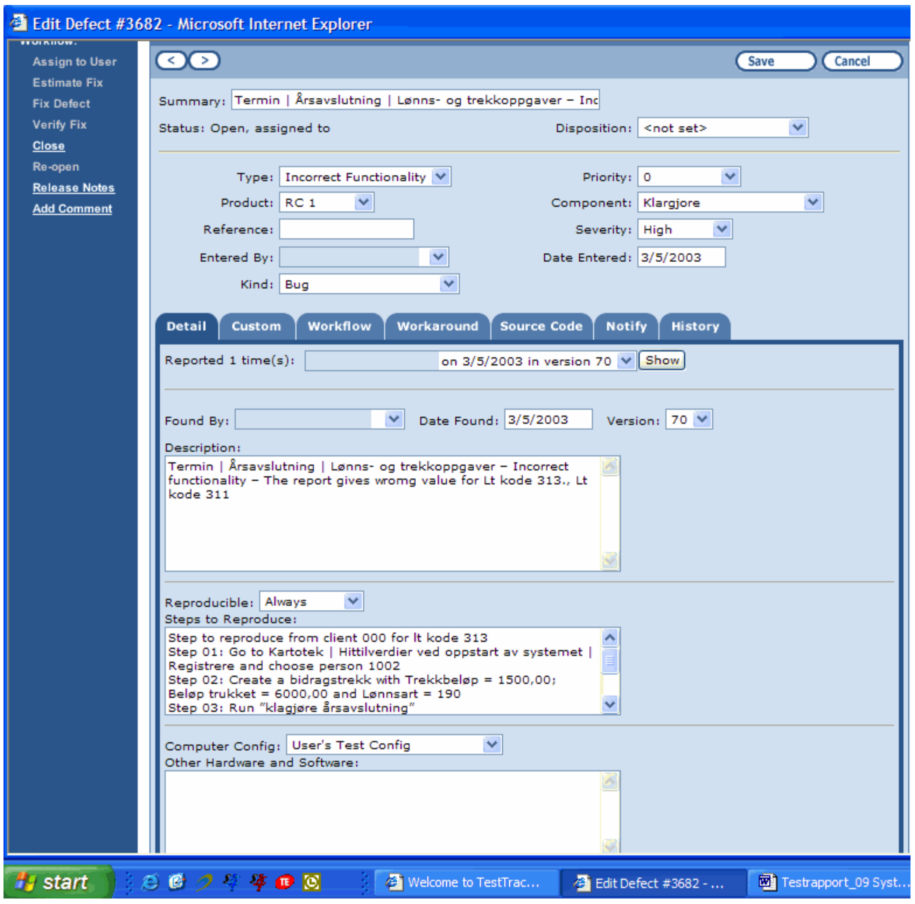
> Almost all the fields were changed due to specific needs in the [SalarySystem] project.
>
> — *RussCo tester I*

Because of this, the following description is a mix of the TestTool default features and the RussCo customizations. In the SalarySystem project the terms “defect”, “bug”, and “error” were used to denote the same thing. In this thesis “bug” is being used consistently.

There are 94 data fields for each bug in TestTool; some are shown in figure 4.7 on the following page while others are shown in other modes. There are around 20 other modes for each bug depending on what action the user wants to perform. For instance, generating a report is performed using a different mode than assigning a bug. The fields are not equally important for understanding the use of TestTool, so a selection of fields found to be most relevant is made.

The bugs chosen *not* to be described in detail can be divided into four groups. The first group of fields enables attachments of screen shots, source code, and other files relevant to a bug. These attachments may help the tester or developer to better understand how to fix or test the bug. The second group of fields contains the names of who performed different actions (these actions are described in more detail below). The third group contains the dates of when these actions were performed, while the last group consists of additional information about the bug. Examples from the latter group are how the bug was fixed, notes from the customer about the verification done by the developer, and the difference between the actual and the estimated time to fix a bug.

The following is a description of the most relevant fields in TestTool. The fields are presented using the sequence they in which they appear in figure 4.7 as a starting point:


Edit Defect #3682 - Microsoft Internet Explorer

Workflow:
Assign to User
Estimate Fix
Fix Defect
Verify Fix
Close
Re-open
Release Notes
Add Comment

Save Cancel

Summary: Termin | Årsavslutning | Lønns- og trekkoppgaver – Inc

Status: Open, assigned to

Disposition: <not set>

Type: Incorrect Functionality
Priority: 0
Product: RC 1
Component: Klargjore
Reference:
Severity: High
Entered By:
Date Entered: 3/5/2003
Kind: Bug

Detail | Custom | Workflow | Workaround | Source Code | Notify | History

Reported 1 time(s): on 3/5/2003 in version 70 Show

Found By:
Date Found: 3/5/2003
Version: 70

Description:
Termin | Årsavslutning | Lønns- og trekkoppgaver – Incorrect functionality – The report gives wromg value for Lt kode 313., Lt kode 311

Reproducible: Always
Steps to Reproduce:
Step to reproduce from client 000 for lt kode 313
Step 01: Go to Kartotek | Hittilverdier ved oppstart av systemet | Registrere and choose person 1002
Step 02: Create a bidragstrekk with Trekkbeløp = 1500,00;
Beløp trukket = 6000,00 and Lønnsart = 190
Step 03: Run "klagjøre årsavslutning"

Computer Config: User's Test Config
Other Hardware and Software:

start | Welcome to TestTrac... | Edit Defect #3682 - ... | Testrapport_09 Syst...

Figure 4.7: A bug as it is represented in TestTool.

Number: Every bug has a unique number which is useful for identification.

Summary: A short text describing the bug. In the SalarySystem project, this description consisted of the navigation path from SalarySystem leading to the bug, for example "Termin | Årsavslutning | Lønns- og trekkoppgaver". This made the bugs easily identifiable.

Status: This field is generated by TestTool. It shows the status of the bug as well as whom it eventually is assigned to. The status can be either Open, Open (Re-opened), Open (Verify failed), Fixed, Closed (Verified), or Closed (see figure 4.7). This field is useful for generating reports of for example how many bugs are open, how many bugs are closed et cetera.

Disposition: The field has the following values:

- Open – Not Reviewed
- Open – Reviewed
- Need Customer Input
- OK to Implement
- Fix In Future Release
- Hold
- Rejected

Bugs in the project marked “Need customer input”, are put into a report and sent to ScanSys together with the weekly report. The ScanSys staff then knew what information RussCo needed.

Type: Describes the kind of bug. The field has the following values:

- Crash – Data Loss
- Crash – No Data Loss
- Incorrect Functionality
- Erroneous Data
- Cosmetic
- Feature Request
- Investigate

RussCo also added the value “GUI” which means that the bug has to do with the graphical user interface. The field is also useful when deciding what priority the bug should be given.

Priority: The value of this field denotes how quickly the bug has to be fixed. In the project, the values 0, 1, 2, 3, 4, and 5 were used. The Russian project manager prioritized the bugs, and the developers and testers fixed or tested the bugs according to the priorities on the bugs assigned to them.

Product: Means what part of the project is being influenced by the bug. One example from the case is “RC1” which means “release candidate 1”.

Component: Means what component in the source code the bug belongs to.

Severity: The developer’s personal opinion about how severe the bug is described by the following values:

- Causes Crash

- No Workaround
- Workaround
- Cosmetic

In contrast to the field "type" which is a more objective description of the severity of the bug, this field allows the developer to give his or her own opinion of the bug.

Entered by: Name of person who entered the bug into TestTool.

Date entered: The date the bug was entered into TestTool.

Version: In the SalarySystem project, the filed was used to denote what build the bug was found on. In comparison to the field called product, this field presupposes several versions between every product release.

Steps to reproduce: This field contains a written description on how to reproduce a bug. The field is useful for developers and testers to find the bug in the system (see also figure 4.7).

Description: Contains a more extensive description of the bug in addition to other relevant information.

The fields described above provide extensive information about a bug with respect to what part of the system it belongs to, what kind of bug it is et cetera. This information provided a common understanding between the two sites.

Actions

Every time an action is performed on a bug, this event is logged by TestTool. This information is stored in a form called "Workflow" which provides an overview over the history of the bug. For every bug, the following actions can be performed:

Register new bug: When a bug is found during testing, it is put into TestTool. TestTool automatically gives the new bug a unique number.

Assign defect: A developer or tester gets a bug assigned to him or her for fixing, testing, or verifying. In the project, the Russian project manager did most of the assignment. Other project members assigned bugs to some extent though.

Estimate fix: After a bug is registered, the amount of work it takes to fix it is estimated. In the project the Russian project manager did most of the estimation. If the developer who gets the bug assigned to him or her for fixing finds the estimate to be wrong, he or she can re-estimate the bug.

Describe fix: After the bug is fixed, information about the fixing is put into TestTool. Examples of fields to be filled in are “Resolution” (how the bug was fixed) and “Hours to fix”.

Verify fix: When a bug is fixed, it is assigned to a developer who verifies it. Dependent on whether the fix was correct, the bug either passes or fails the test. If it passes, the status is set to “closed”. If it fails, the status is set to open and the bug must be reassigned.

Close defect: A bug can be closed for several reasons, having any status except from “closed”.

Re-Open defect: A closed bug can be re-opened which gives it the status “Open”.

Release Notes: The release notes are the basis for readme files and other documentation about the project.

Add comments: Comments can be added to any bug. Such comments may concern details relevant to the bug.

Figure 4.8 on the next page, based on the interviews of staff in ScanSys and RussCo, describes the workflow in the SalarySystem project as the actions described above were performed. Most of the bugs found in Norway were put into TestTool by the Norwegian test coordinator, while in Russia the testers in the test department registered most of the bugs. The process from a bug was found until it was fixed and closed sometimes took up to two weeks.

When a bug was found at the Norwegian test department, the tester sent an e-mail to the Norwegian test coordinator describing the bug. When the ScanSys testers tested, they had two PCs running; one with the old SalarySystem, and one with the redesign version. The testers’ job was to report any difference between the old and the new version. Because not all of the differences they found were bugs, this decision was made by the Norwegian test coordinator. Examples of differences not being bugs are minor changes to the GUI, or improvements made by the Russians caused by unsatisfying performance in the old system.

When a bug was found at the Russian test department, the bug was put into TestTool by the tester who found it. When a bug was put into TestTool, an e-mail was automatically sent to the project mailing list. This also happened when someone at the Norwegian site registered new bugs, and increased the Norwegians’ knowledge about the work progress

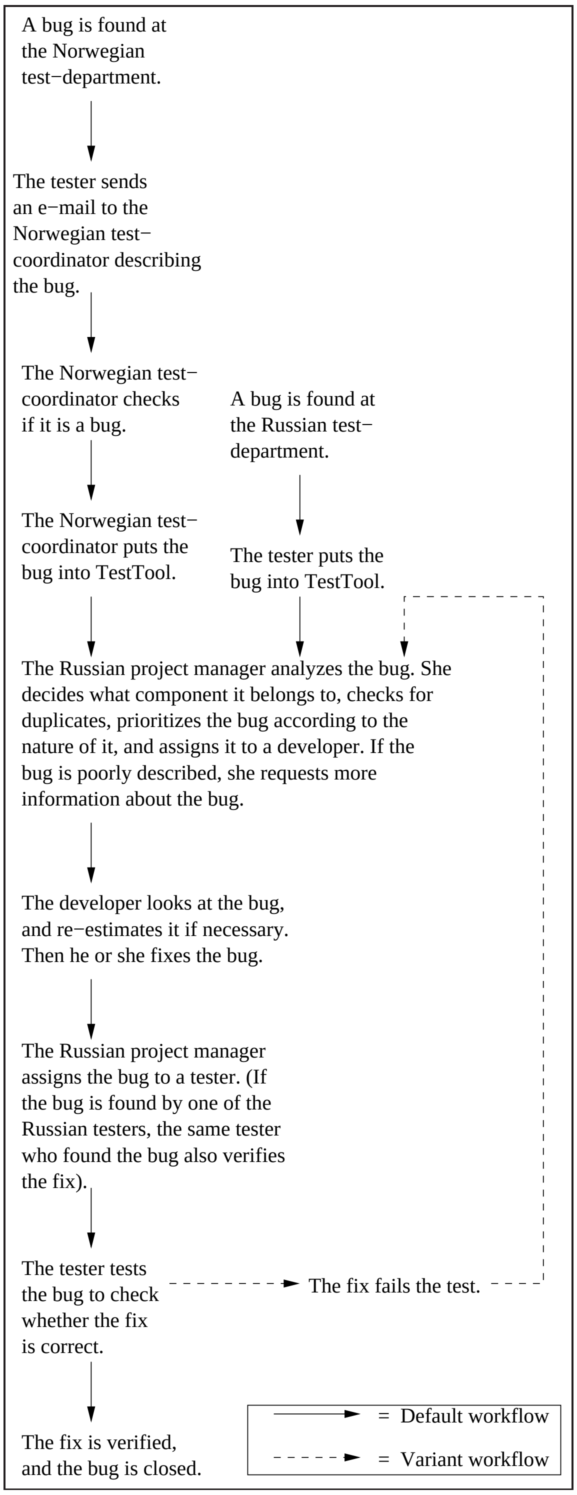

A bug is found at the Norwegian test−department.
The tester sends an e−mail to the Norwegian test−coordinator describing the bug.
The Norwegian test−coordinator checks if it is a bug.
A bug is found at the Russian test−department.
The Norwegian test−coordinator puts the bug into TestTool.
The tester puts the bug into TestTool.
The Russian project manager analyzes the bug. She decides what component it belongs to, checks for duplicates, prioritizes the bug according to the nature of it, and assigns it to a developer. If the bug is poorly described, she requests more information about the bug.
The developer looks at the bug, and re−estimates it if necessary. Then he or she fixes the bug.
The Russian project manager assigns the bug to a tester. (If the bug is found by one of the Russian testers, the same tester who found the bug also verifies the fix).
The tester tests the bug to check whether the fix is correct.
The fix fails the test.
The fix is verified, and the bug is closed.
= Default workflow
= Variant workflow

Figure 4.8: The process of fixing a bug.

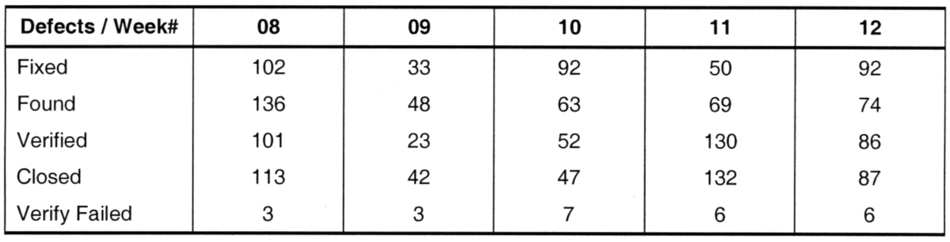
in Russia. This overlooking was especially important in Norway after the near-breakdown of the project when the Norwegian project members started questioning the Russians' ability to finish the project.

All new bugs were analyzed by the Russian project manager. She checked for duplicates, filled in additional information about the bug, prioritized it, and assigned it to a developer. If the bug was poorly described, she requested more information.

When a bug was properly described, the Russian project manager assigned it to a developer. The two main criteria for this selection were availability and skills. Then the developer fixed the bug, and set the value of the "Status" field to "Fixed". After the bug was fixed, the Russian project manager assigned it to a tester for verification. If the bug was initially found by a Russian tester, the same tester also verified the fix. A bug either passed or failed the verification. If it passed, the bug was closed. If it failed, the bug was reassigned to a developer for fixing and the same cycle was repeated.

Reporting

Another feature of TestTool is the reporting function. TestTool provides a set of report templates with information about the bugs and the bug correction process. It is also possible for users to design their own reports containing any data from the TestTool database. Every week during the project, a report was prepared by RussCo and sent to ScanSys. This report contained information about project status, planned activities, and issues that needed input from the customer. Much of the contents in the weekly report were generated by TestTool. Figure 4.9 shows the distribution of different kinds of bugs during the weeks before the report was issued.



Defects / Week#	08	09	10	11	12
Fixed	102	33	92	50	92
Found	136	48	63	69	74
Verified	101	23	52	130	86
Closed	113	42	47	132	87
Verify Failed	3	3	7	6	6

Figure 4.9: Part of the weekly report. Data generated by TestTool.

Since the weekly report enabled the Norwegians to monitor the work progress, this feature increased the ScanSys knowledge about the work taking place in Russia.

Once a week, RussCo also generated a report from TestTool which contained all bugs where the field “Disposition” was set to “Need Customer Input”. This report was issued together with the weekly report, and had two kinds of bugs: (1) bugs found by the Russians, but which they did not fully understand, and (2) bugs found by the Norwegians, but which were poorly documented. Figure 4.10 on the next page shows two bugs from the “Need Customer Input” report. When ScanSys got the report, they prepared an answer and replied using e-mail. When RussCo found the reply to be sufficient, the status of the bug in the report was changed to “Clarified”.

Because the reports generated by TestTool enabled a common understanding on a detailed level between the two companies, it became easier to share relevant information between the sites.

4.5.2 The role of TestTool

In the beginning of the SalarySystem project, TestTool was put into use. But because the Russian software deliveries were incomplete with respect to functionality and GUIs, the use of TestTool was found not to be useful and thus abandoned:

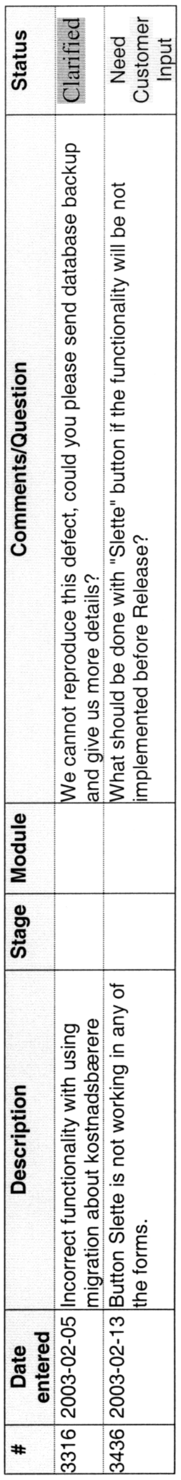
> There were so many bugs in the redesigned source code in the beginning so we could not test for more than a couple of hours before it all broke down, so TestTool was not useful. At this time, our frustration was big.
>
> — *ScanSys project manager*

Frustration was also present at the Russian site where the developers tried to increase their knowledge of SalarySystem:

> We tried to understand the business logic from the source code, but did not make it.
>
> — *RussCo developer I*

Trust in Norway

Four months after project start a new attempt to use TestTool was made. At this point, the software deliveries from Russia still contained many bugs, but the Russians did not register them in TestTool. Figure 4.11 on page 74 shows the number of bugs reported in TestTool, and how these numbers corresponded with the key events of the project. These events influenced the number of bugs registered in TestTool, and thus



#	Date entered	Description	Stage	Module	Comments/Question	Status
3316	2003-02-05	Incorrect functionality with using migration about kostnadsbærere			We cannot reproduce this defect, could you please send database backup and give us more details?	Clarified
3436	2003-02-13	Button Slette is not working in any of the forms.			What should be done with "Slette" button if the functionality will be not implemented before Release?	Need Customer Input

Figure 4.10: The bugs that need customer input.

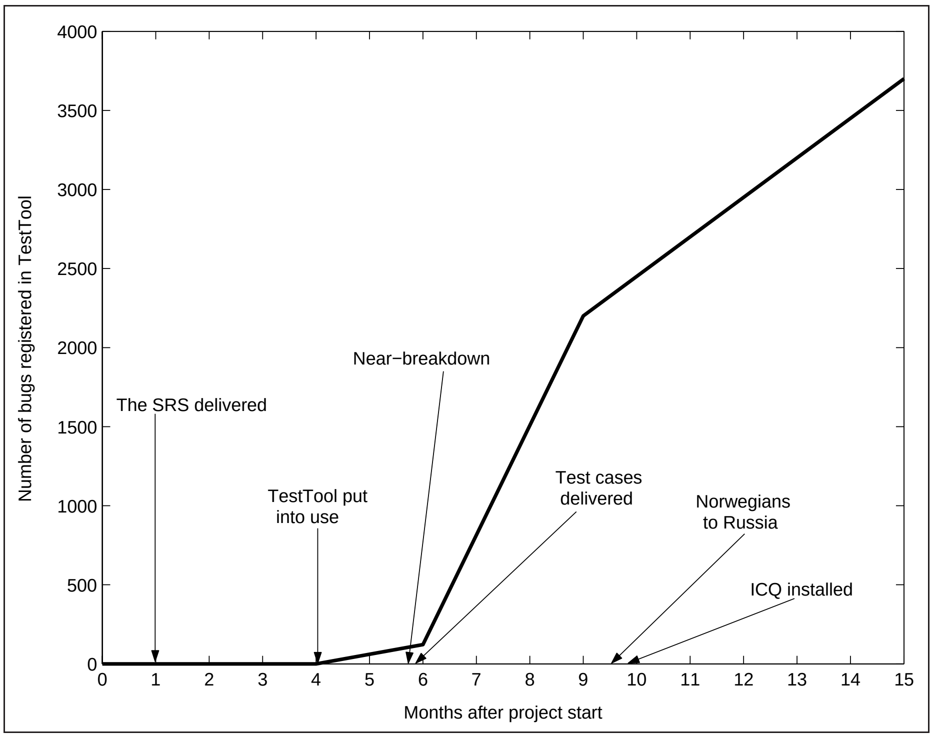


Figure 4.11: Number of bugs registered in TestTool compared with the key events in the project.

made the project members—both in Norway and Russia—feel more or less confident about the project.

In many software projects, few bugs would probably have been considered positive. But since the Norwegians found many bugs in the software deliveries, few bugs in TestTool were seen to indicate a low degree of domain understanding in Russia. However, when the number of bugs later in the project increased, the Norwegians once again felt confident about the Russians' competence:

> Because of the increase in bugs reported, I felt safer about the bug correction done by the Russians.
>
> — *ScanSys test coordinator*

Since the use of TestTool increased the Norwegians' confidence about the SalarySystem project, TestTool is seen as a structural mechanism. Other attributes were also influenced though. First, as described above, the competence-based trust was influenced by the number of bugs reported. Second, TestTool also increased the Norwegians' knowledge about the development in Russia which, in turn, influenced the degree of knowledge-based trust. Third, because of this increased knowledge, the calculation-based trust was also influenced. For instance, when the Norwegians started questioning the Russian competence, the perceived opportunity of the ScanSys–RussCo relationship would necessarily be influenced too. Fourth, since knowledge-based, competence-based, and structure-based trust all has to do with predictability, this attribute of trust was also influenced. For instance, section 2.2.3 on page 23 describes two ways of increasing predictability: (1) enable monitoring, and (2) influence the behavior of the partner. Considering (1), TestTool *did* increase the ability to monitor the development in Russia, and (2) enabled the Norwegians to better describe SalarySystem which influenced the Russian development, so the predictability increased too. However, as described above, the use of TestTool both increased and decreased the degree of trust during the project. This is in accordance with the description of predictability where the predictable behavior not necessarily is the preferred behavior.

One example of how TestTool improved the predictability is an event that happened at the end of the SalarySystem project. Because the Russians tested using automated tests, they did not find bugs in the SalarySystem GUI. Automated tests means testing by putting data from a backup file into the system instead of entering data as end-users. The Norwegian test coordinator became aware of this kind of testing by looking at the kinds of bugs in TestTool. Because of the detailed information about every bug registered in TestTool, he was able to see that the Russian testing did not reveal all kinds of bugs. When he confronted the Russian project manager with this, she told about the automated tests, and they agreed that the Russians should test as end-users like the Norwegian test department did. Later, when bugs from the GUI registered in Russia appeared in TestTool, the Norwegian test coordinator got the impression that the project was moving in the right direction.

Trust in Russia

TestTool as a structural mechanism also influenced the RussCo degree of trust in the Norwegians. During the first months of the project, the Russians were frustrated because they did not understand SalarySystem. But when TestTool was put into use, it enabled a more detailed feedback from the Norwegians.

As described earlier, the “Need customer input” document—generated by TestTool—was sent to Norway every week. This document made it easier for the Russians to tell what parts of the system they needed to have explained. And since the Norwegians suddenly had a detailed request for what the Russians needed to have explained, they were able to give the right explanations. This was important because the Russians lacked trust in the Norwegian competence too. The following was said about how the Russians considered the Norwegian competence early in the project:

> [The ScanSys developer] does not have much domain knowledge to transfer.
>
> — *RussCo general manager*

When the Norwegians suddenly gave adequate answers to the Russians' questions, the Russians' competence-based trust in the Norwegians increased. Also, the Russians' faith in a successful project increased which, in turn, built predictability-based trust.

4.5.3 Violated expectations

When the Norwegians' trust in the Russians' competence deteriorated, TestTool was important as described above. However, other events in the SalarySystem project were important too. When the Norwegians suggested the project was running late, they did so because they tried to save the project from a total failure. But because the Russians did not admit to the problems, and later asked for six more months, the Norwegians felt they were being fooled:

> When the Russians did not admit the project was running late, it turned the situation upside-down.
>
> — *ScanSys support coordinator*

The situation made the Norwegians unwilling to go for a compromise, and instead they blamed RussCo for the delay. From the description of betrayal, an action both has to violate the expectations of the trustor *and* be voluntary to be called betrayal. According to the quote above, the Norwegians' expectations about the project were violated. Deciding whether the action was voluntary, on the other hand, requires a more thorough discussion.

From the Russian side, the advantage of not telling the Norwegians about the possible delay would be none. In the beginning of the project, the intention of not telling about their lack of understanding, as described in section 4.4.2 on page 60, was to get the contract. But after the contract was signed, such a motivation was no longer present. According to the contract of the project, the project could be stopped after any finished phase, so there would be no calculation-based argument for the Russians not to tell about the possible delay. Furthermore, betraying ScanSys would mean the end of the project, and thus no opportunities for further contracts. Instead, this thesis suggests that the near-breakdown took place because the Russians lacked the appropriate knowledge about SalarySystem. This lack of knowledge also included knowledge about their own degree of knowledge:

> We should have understood [one month after project start] that we could not finish the project in time because we did not have the people.
>
> — *Former RussCo project manager*

Once again, the possible betrayal is instead seen to be a result of other factors more than an intention to betray the partner. Still, during the near-breakdown the Norwegians' degree of competence-based trust was low.

4.5.4 Unexpected behavior

When the former RussCo project manager was removed at the near-breakdown meeting in Russia, the Norwegians who attended the meeting were surprised. They were not used to such behavior, neither in Norway nor Russia. Furthermore, such behavior was not preferred because the reaction was considered too strong, and the blame was put on one person.

Since predictability stems from observing patterns of behavior and expecting them to continue, the removal of the project manager was a break in these patterns because the situation was unexpected. However, the episode only happened once, thus, it did not become the expected pattern of behavior. Still, the episode made a strong impression on the Norwegians, so this thesis interprets the Norwegians to more likely expect unanticipated episodes in the future. In figure 4.12 on the next page, the result of the situation is illustrated. The Norwegians are found to have had a medium degree of information about the Russians (because of interaction), and a low degree of predictability-based trust because unanticipated episodes in the future suddenly became more likely.

The episode is also connected to knowledge-based trust since the Norwegians through the meeting had first-hand experience of the Russians' behavior. However, as the interaction did not build trust, it is important to notice that not all knowledge about the partner in a GSO relationship builds trust.

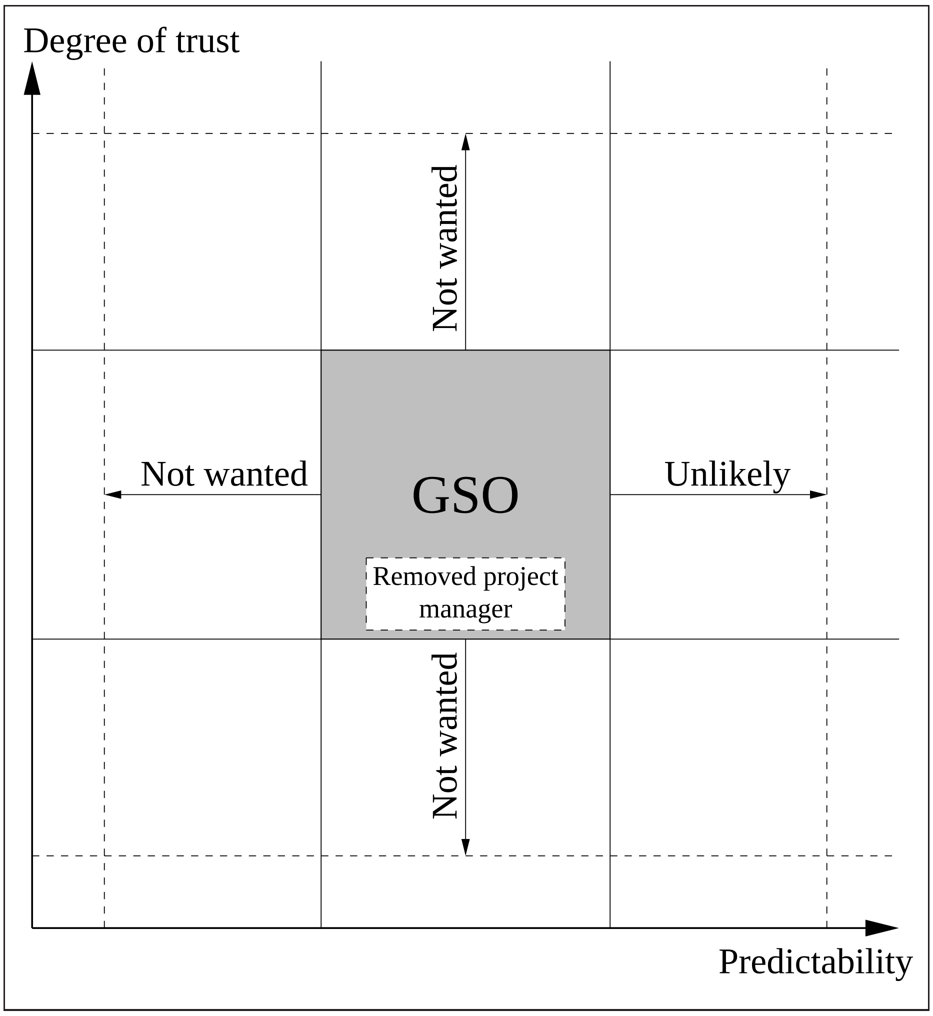


Figure 4.12: The removed project manager.

4.6 The recovery phase

After the near-breakdown, the performance of the SalarySystem project improved. During this phase, several changes were made which all contributed to the positive trend. These changes are discussed in this section focusing on their relation with trust.

One of the positive changes was the introduction of the test cases—documents describing the use of SalarySystem. These documents were prepared by the Norwegians and sent to Russia five months after project start. The test cases improved the knowledge sharing in a number of ways, which helped the Russians understand SalarySystem better. Altogether there were made 15 test cases.

4.6.1 Contents of the test cases

Each test case describes a scenario or operation considered difficult. Some examples of difficult scenarios are the tax and salary rules related to sailors, foreigners living in Norway, or Norwegians living at Svalbard. People from these groups follow different tax and salary rules, and the business logic encapsulated into the source code can make it difficult to understand. Even the highly skilled SalarySystem support staff at ScanSys sometimes found the business logic difficult to understand due to its complex nature:

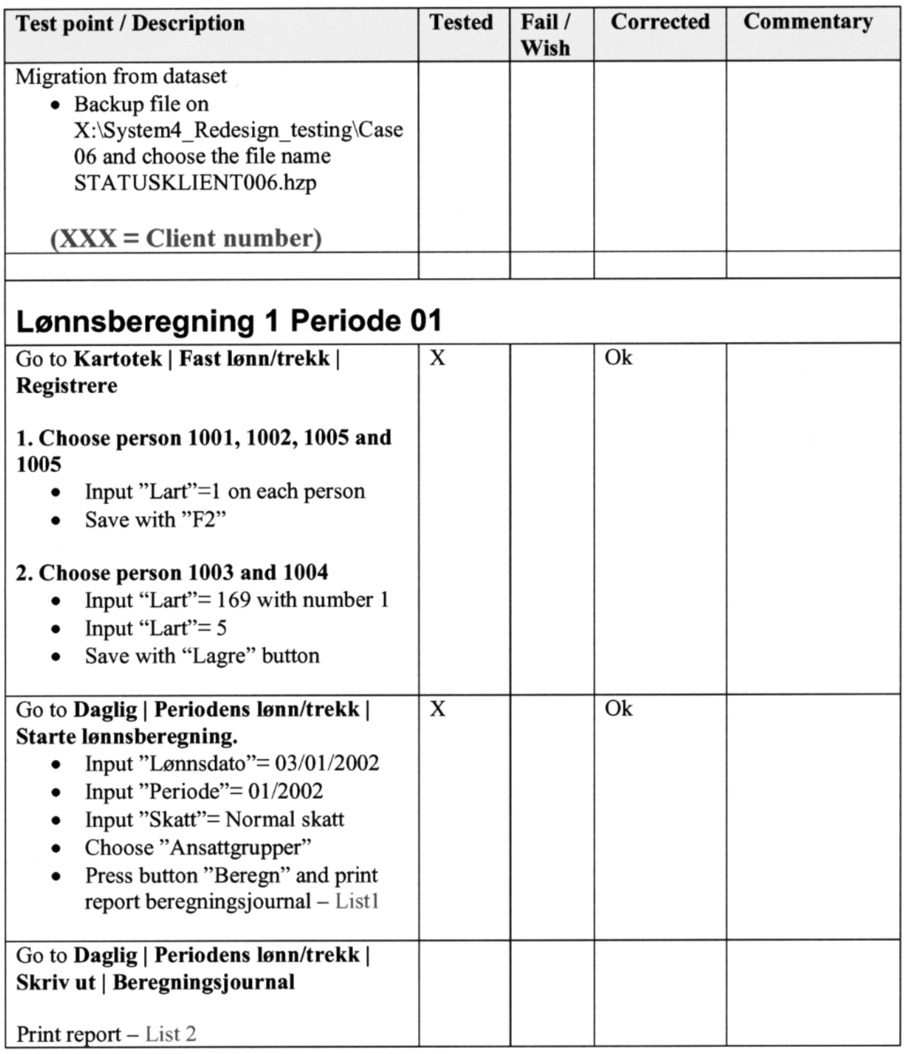
> Don't mix sense and logic when it comes to tax rules.
>
> — *ScanSys support coordinator*

Several operations in SalarySystem may also be difficult to perform because the Norwegian tax and salary rules are encapsulated into SalarySystem, and consequently mirrored in the use of the system. Some examples are import or export of data, tasks related to the annual balance of accounts, or salary calculation. The test case in figure 4.13 on the following page, considers the latter operation.

In addition to a thorough description about how to perform different tasks in SalarySystem, the test cases provide the input and the corresponding output of a specific task. These values are stored in a separate file.

As the test case in figure 4.13 shows, the use of different parts of SalarySystem is described in detail with respect to what buttons to push, what values to fill in and so on. In the given test case, 63 different operations concerning printing of reports, performing a salary calculation, generating a backup file and so forth is described. The operations are not directly connected, and can be performed independently. The sequence of the operations describes the most common way of using SalarySystem.

One important aspect of the use of SalarySystem is that every year is divided into periods. Each period is closed when the salaries are being paid. Because the use of SalarySystem—and the system itself—is highly influenced by this division, the test cases are divided into periods too. Every test case consists of four periods, each describing approximately the same operations. However, the differences between the parts are necessary in order to simulate the use of SalarySystem during a whole year. For example, in Norway employees pay half tax in December, so the last period of the test case would consequently describe a salary calculation using half tax.



Test point / Description	**Tested**	**Fail / Wish**	**Corrected**	**Commentary**
Migration from dataset • Backup file on X:\System4_Redesign_testing\Case 06 and choose the file name STATUSKLIENT006.hzp **(XXX = Client number)**				
Lønnsberegning 1 Periode 01				
Go to **Kartotek | Fast lønn/trekk | Registrere** **1. Choose person 1001, 1002, 1005 and 1005** • Input ”Lart”=1 on each person • Save with ”F2” **2. Choose person 1003 and 1004** • Input “Lart”= 169 with number 1 • Input “Lart”= 5 • Save with “Lagre” button	X		Ok	
Go to **Daglig | Periodens lønn/trekk | Starte lønnsberegning.** • Input ”Lønnsdato”= 03/01/2002 • Input ”Periode”= 01/2002 • Input ”Skatt”= Normal skatt • Choose ”Ansattgrupper” • Press button ”Beregn” and print report beregningsjournal – List1	X		Ok	
Go to **Daglig | Periodens lønn/trekk | Skriv ut | Beregningsjournal** Print report – List 2				

Figure 4.13: Part of test case number six.

4.6.2 The role of the test cases

The test cases were helpful in several ways: First, they made the Russians find the “hidden forms”. Because of the Norwegian tax and salary rules, some forms in SalarySystem are displayed only when specific data are put into the system. For example, employees being more than 62 years of age, follow special tax and salary rules. The user has to register such persons in order to access the “hidden forms”, so these values are given in the test cases. Second, the test cases helped the Russians to find any bugs between the input and the output. That is, since the Russians tested using automated tests to a large extent, it made them miss bugs in the interface. When they got the test cases, their attention turned against testing as end-users which generated more bugs. The lack of understanding about the use of SalarySystem was one reason for this until the test cases gave them a more integral understanding of the system. Third, the test cases helped the Russians trace the logical sequence of use of SalarySystem. Looking at figure 4.13, such a sequence could be to perform a salary calculation before printing the report.

The test cases increased the Russians' understanding of SalarySystem and the business logic encapsulated into it. That was positive in several ways because the limited amount of understanding about the SalarySystem project was a problem for the Russians' ability to make the project successful:

> If we had got the test cases earlier, we would have been able to make more accurate estimations.
>
> — *RussCo project coordinator*

Since the test cases increased the Russians' understanding of SalarySystem and its embedded business logic, this thesis suggests that competence-based trust was built both in Norway and Russia because of the documents. When the software deliveries from Russia were improved with respect to functionality and finished GUIs, the Norwegians felt good about the Russians' knowledge about SalarySystem, and their competence-based trust in the Russians increased. However, the Norwegians' understanding of the Russians' increased competence was also due to TestTool where they could monitor the progress of the work in Russia.

At RussCo, the detailed test cases increased the Russians' impression of the Norwegian knowledge about SalarySystem because they suddenly got adequate explanations on problems they had. Thus, the Russians' competence-based trust in the Norwegians increased as a result of the test cases too.

The test cases also fit the description of a structure, and considering the importance of the documents in the project, this thesis interprets the test cases to have built structure-based trust in Russia. From the description of structure as an attribute of trust, the test cases can be used to influence the behavior of the partner. For instance, the Norwegians—by sending the test cases—explicitly taught the Russians how to understand SalarySystem which, in turn, influenced the development in Russia.

Considering the test cases as structural mechanisms, one question to ask is whether the degree of structure was appropriate during the project. Section 2.2.3 on page 23 identifies lack of structure as equal to excessive faith in trust. Because of the problems the Russians had understanding the use of SalarySystem during the early phases of the project—with poor effort made to help them—the Norwegians' faith in the Russians' competence was excessive. Also, the Russians would have preferred to get the test cases earlier. When the Russians got the test cases, their understanding increased which was positive for the project. Thus, in this thesis, the degree of structural mechanisms is interpreted to have been too low until the near-breakdown and appropriate after the test cases were introduced. The test cases being a structural mechanism was guarding against lack of competence in Russia.

4.6.3 Other positive changes

In addition to the introduction of TestTool and the test cases, several changes were made in the SalarySystem project which influenced the degrees of trust in Norway and Russia. These changes are discussed below in relation to trust.

When the Norwegians visited Russia ten months after project start, the meeting had several advantages. During the meeting, the Norwegians met the testers and developers in RussCo for the first time. Prior to the meeting, the project members only knew the names of their colleagues. This interaction was important for knowledge-based trust to appear between the individuals in the project. For instance, the ScanSys test coordinator was more confident the Russians could finish the project after the face-to-face meeting. Thus, his predictability about the project was also influenced. During the meeting, the Norwegians also got a better chance to explain details about SalarySystem and its embedded business logic to the Russians which increased the Russians' understanding. As a consequence, the Russians' impression of the Norwegian knowledge about SalarySystem increased too. This bidirectional way of building trust is quite similar to how the test cases influenced the degree of trust.

In Norway, knowledge-based trust was also built as a result of the improved reporting from Russia during the recovery phase. These reports—partly generated by TestTool—contained information about the work taking place in Russia. Because the reports described the work in Russia in a detailed way, this thesis interprets the reports to have increased the knowledge-based trust in Norway. Competence-based trust was built too because the Norwegians could monitor the progress of the development in Russia.

4.7 Interdependency

In the beginning of the SalarySystem project, ScanSys was the strong part operating in a buyer's market. RussCo, on the other hand, had a strong calculation-based trust in the relationship. However, as the project went by, the progress of the SalarySystem project escalated and the knowledge of SalarySystem and its embedded business logic in Russia increased. As a result, ScanSys found itself getting more dependent on RussCo. In addition, the Norwegians started to consider RussCo as a reliable and competent software supplier useful in future software projects. Using the attributes of trust, this shift in the relationship can be looked upon as ScanSys had got calculation-based trust in RussCo. The development of such trust was in contrast to how the Norwegians' calculation-based trust prior to the SalarySystem project was in GSO as a phenomenon more than RussCo as a future business partner. After the problems related to communication, knowledge sharing, project reporting, and shift of staff, ScanSys found the relationship had become advantageous. The episode where ScanSys decided to finish phase eight of the project internally, but still pay RussCo \$14 000 according to the contract, is in this thesis interpreted as an example of how ScanSys had got calculation-based trust in RussCo. Instead of claiming their right and let RussCo finish the eighth phase—which might had been advantageous in the short term—they found the other solution to be better in the long run. Sahay, Krishna, and Nicholson (2003) suggest that calculation-based trust is strongly associated with the early stages of the relationship. However, in the ScanSys–RussCo relationship such trust is seen to be present more than two years after the relationship was formed.

Since the calculation-based trust was connected to the faith ScanSys had in RussCo as a competent software supplier, competence-based trust was important too. That is, the Russians' competence was one of the reasons the Norwegians had got calculation-based trust in RussCo, so the competence-based trust *influenced* the degree of calculation-based trust. Moreover, the improved interaction across the sites increased the Norwegians' degree of knowledge about the Russians as described in the previous section. This knowledge is seen as one of the reasons for the Norwegians' calculation-based trust too.

From the descriptions of the attributes of trust in chapter 2, competence and knowledge were seen to increase predictability. That is, if (at least) one these attributes are present, predictability is more likely to emerge. In addition, the weekly reports—considered structural mechanisms—increased the predictability too. At last, calculation and predictability being quite similar, they influence each other as well. The relationships between the kinds of trust being present at ScanSys in the interdependency phase is illustrated in figure 4.14 on the following page.

The arrows in the figure illustrates how the attributes of trust influenced each other.

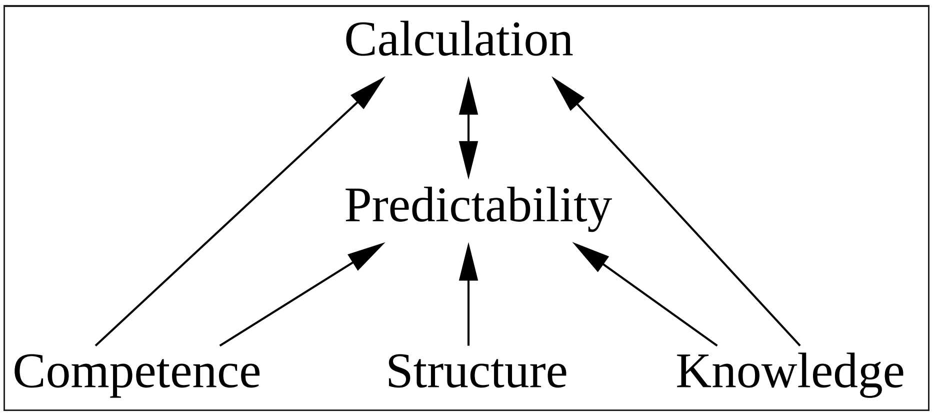


Figure 4.14: The influences between the attributes of trust being present in the interdependency phase.

Chapter 5

Discussion

In this chapter, the findings from chapter 4 form the background of the discussion of the main objective in this thesis:

> **Identify and describe trust in Global Software Outsourcing relationships.**

The chapter is divided into two parts. The first part discusses the findings about trust while the latter looks at how this new understanding can be used to suggest ways to perform successful GSO projects. These two parts thus reflect theoretical and practical implications respectively.

5.1 Findings about trust

In the light of the findings from the previous chapter, a legitimate question to ask is whether and how is trust needed in GSO projects? Based upon these findings, the suggested answer is *yes*. For instance, when the project almost failed during the near-breakdown phase, the Russians had a strong calculation-based trust. This trust made them offer to finish the project without being paid for additional time although they did not have to make this offer according to the contract. Thus, without trust, the project would most likely have failed. From the Norwegian point of view, their gut feeling during the first visit at RussCo—interpreted as knowledge-based trust—was an important reason for the ScanSys–RussCo relationship to be initiated in the first place.

Another finding from the previous chapter is that the attributes of trust were present in all phases of the SalarySystem project. Furthermore, all attributes apart from betrayal were identified. That is, none of the findings suggested that betrayal was breaking trust or *not* betraying was building trust. Thus, the attribute is removed from the list of attributes of trust. Still, betrayal is considered a threat to GSO projects, so it is important to be aware of such behavior. In the SalarySystem project, both ScanSys and RussCo had their chance of potentially betraying the partner. For instance, ScanSys could have failed to pay as agreed upon in the contract, and RussCo could voluntarily have assigned too few people to the project in order to save money. This thesis suggests that because of the high degrees of calculation-based trust on both sides during the SalarySystem project, betraying the partner was less likely to happen. Thus, calculation-based trust is seen as a way of guarding against such behavior.

At last, the SalarySystem project illustrated how calculation-based trust not only was associated with the early stages of the relationship, but was still present more than two years after the relationship was formed.

These findings about trust are implicit when the attributes of trust are discussed in the next sections.

5.1.1 How the attributes address the factors

In chapter 2, the factors that had to be addressed by the attributes were presented. According to these, trust was seen to be multidisciplinary, present within *and* between organizations, multilevelled, related to humans *and* systems, changing over time, having varying degrees, being bidirectional, and socially constructed. Since these factors were considered necessary to be addressed for the attributes of trust to have explanatory power in the GSO context, another evaluation—based on the findings from chapter 4—is provided below. The findings are illustrated with examples from the SalarySystem project.

Trust is *multidisciplinary*, so the attributes of trust draw upon ideas from sociology, economics, and psychology in order to address the diverging views of trust considered relevant in the GSO context. In the SalarySystem project, ideas from the different disciplines were found to be relevant in the following way: First, the structure attribute, based upon ideas from sociology, was used to explain why mechanisms like the SRS, TestTool, the weekly reports, and the test cases were important in the project. Second, the idea of calculation from the economic literature was used to explain why the SalarySystem project was initiated. At last, goodwill from the psychology literature was used to understand how the Norwegians interpreted the first meeting at RussCo.

This thesis looks at trust *within and between organizations*, and the findings from chapter 4 suggest that trust is important in both these environments. For instance, the calculation-based trust the Russians had in ScanSys was present in addition to how trust—or lack of trust—was important within RussCo when the Russian project manager was removed during the near-breakdown.

Trust being different regarding what *organizational level* it is studied on, was found to be addressed when the attributes were applied on the SalarySystem project. For instance, when the RussCo general management had calculation-based trust in getting the SalarySystem contract, the RussCo developers worked hard in order to make the Norwegians build competence-based trust.

Trust coming from both *systems and humans* were addressed by the attributes too. As described above, structural mechanisms like the weekly reports, the SRS, TestTool, and the test cases were important when building trust in addition to the Norwegians' competence-based trust in the "Delphi Guru".

Figure 5.1 summarizes the results from chapter 4 concerning trust in Norway. The figure presents the results in a schematic way compared to the description of trust from the previous chapter. This is done in order to illustrate tendencies about trust in the SalarySystem project. As the figure shows, trust in the project *changed as time went by*. For instance, ScanSys had calculation-based trust in GSO as a phenomenon before the SalarySystem project was launched, but later this trust changed into calculation-based trust in RussCo.

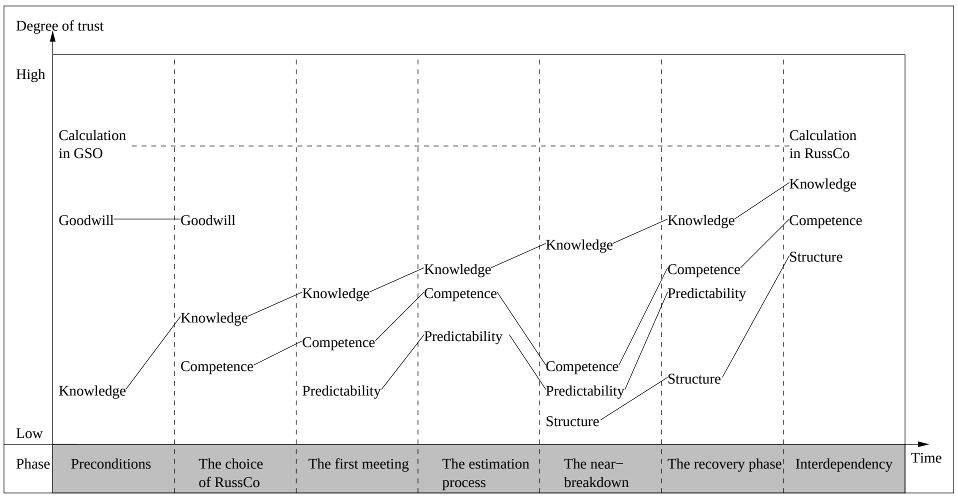


Figure 5.1: Degrees of trust in Norway during the SalarySystem project.

In addition to trust changing over time, trust is also seen to have varying *degrees*. That is, the trustee can trust little or much. For instance, the Norwegians' degree of competence-based trust decreased during the near-breakdown before it reappeared later in the project.

Figure 5.2 on the following page summarizes the results from the previous chapter concerning trust in Russia. Comparing the figure with figure 5.1, trust was different at the two sites with respect to what attributes were important and their associated degrees. Thus, different kinds of trust were built at the two sites as a result of the same events which make trust *bidirectional*. One example is how the initiation of the SalarySystem project built calculation-based trust in Russia while in Norway goodwill trust was more important during this time. Also, the way ScanSys was the strong part during the early stages of the project influenced the trust.

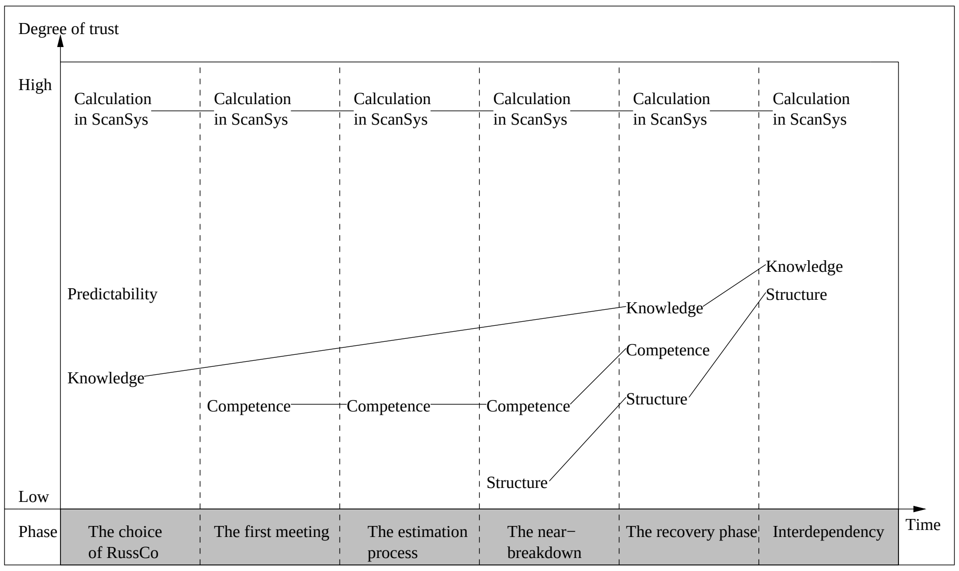


Figure 5.2: Degrees of trust in Russia during the SalarySystem project.

At last, trust being *socially constructed* was found in the project too. For instance, the first time the Norwegians went to Russia, the interaction with the Russians built the Norwegians' trust.

As seen above, all the factors were addressed when the attributes of trust were applied to the SalarySystem project. Thus, with respect to the factors, the attributes are seen to be useful for understanding trust in GSO relationships.

5.1.2 Influence between the attributes

According to the findings from chapter 4, the attributes of trust influenced each other in several ways. Based upon these findings, this thesis suggests figure 5.3 on the next page as a model to identify and describe trust in GSO relationships. The figure shows how the attributes of trust are seen to influence each other. The enumeration indicates only the corresponding descriptions. That is, they do *not* indicate in what sequence trust appears. The arrows in the figure are described and illustrated by examples from the SalarySystem project below:

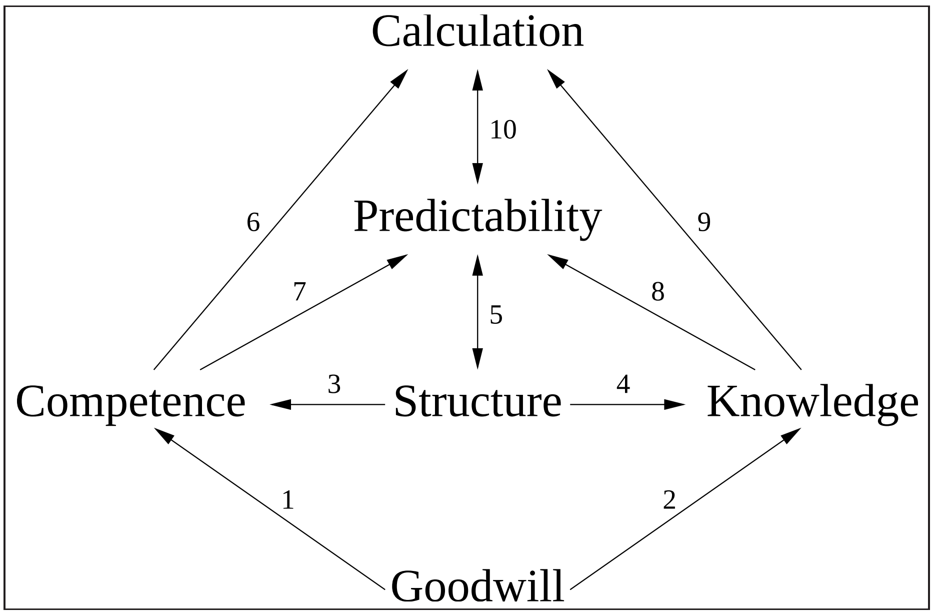


Figure 5.3: The relationships between the attributes of trust.

1. Goodwill influences the way individuals understand the competence of the partner. In the SalarySystem project, the Norwegians' strong belief in the competence of the "Delphi Guru" made them believe the Russians' knowledge about SalarySystem and its domain was adequate to develop the new system.
2. Goodwill also influences the way the knowledge of the partner is understood. Since knowledge about the partner is based upon interaction, the goodwill towards the trustor will influence whether the interaction is interpreted in a positive way. When the Norwegians visited RussCo for the first time, they interpreted characteristics of RussCo in a positive way.
3. Structural mechanisms influence how competence trust is built. In The SalarySystem project, the use of TestTool helped the Norwegians to answer the Russians' question which built competence-based trust in Russia.
4. Similar to arrow 3, structural mechanisms can also influence how knowledge trust is built. In The SalarySystem project, the use of TestTool increased the Norwegians' knowledge about the software development in Russia.

5. Structural properties like reporting mechanisms and rules for response time increase predictability. Also, the degree of predictability is seen to influence the structural properties. In the SalarySystem project, the weekly report from Russia influenced the Norwegians' predictability about the SalarySystem project. Furthermore, one of the reasons the weekly reports were improved after the near-breakdown, was because the degree of predictability was considered too low.

6. The competence of the partner influences how the opportunities and risks of a GSO project are calculated. In GSO relationships, getting access to competent developers is sometimes an important reason for the GSO project to take place. Thus, the company considers competence as an argument when calculating whether the relationship is advantageous. During the recovery phase, the progress of the project built the Norwegians' calculation-based trust in the Russians.

7. The competence of the partner influences how a GSO relationship is predicted. When the Norwegians evaluated the SRS, they made the prediction that the Russians possessed skills about SalarySystem.

8. The knowledge about the partner influences how a GSO relationship is predicted. The Norwegians recognized characteristics like a Western organization style which increased their feeling of predictability about future events.

9. Knowledge about the partner will influence the assessment of associated risks and opportunities. Prior to the SalarySystem project, the Norwegians' knowledge about GSO as a phenomenon built calculation-based trust.

10. Since calculation and predictability are quite similar, they influence each other. Calculation of the risks and opportunities associated with the relationship is a way of predicting the future as well as predictability being an aspect of calculation. The Norwegians' increased predictability about the SalarySystem project in the interdependency phase influenced their calculation-based trust in the Russians. Also, the Russians' strong calculation-based trust during the project influenced their feeling of predictability.

Evaluation

As figure 5.3 shows, the attributes of trust influence each other in several ways. That is, if one attribute of trust is present, the other attribute is more likely to appear. Still, all the attributes can appear independently.

The figure does not explicitly tell in what sequence the attributes appear, rather how they mutually influence each other. Yet, some attributes are associated with the early stages of the relationship. For instance, goodwill—which has to do with personality—is present before the project starts. However, if the individual has a low tendency to trust others, other attributes will be more important when building trust. Furthermore, calculation-based trust is often associated with the initial phases of a relationship, but according to the findings in the SalarySystem project, this attribute of trust was also present later in the project. As seen in the figure, calculation-based trust appears more likely if the other attributes are already present. Still, being on top of the figure does *not* suggest that this kind of trust appears late in the relationship. Yet, since calculation-based trust appears as a result of knowledge-based trust which, in turn, appears after long-time interaction, the figure explains why calculation-based trust is also seen to be associated with more mature relationships.

There are more arrows that can be thought of, for instance how goodwill influences calculation. However, the figure is based upon findings from the SalarySystem project where no such connection was identified. Thus, the attributes and the influences between them are considered *tendencies* of how the degrees of trust can increase or decrease.

Since all the factors about trust are addressed by the attributes of trust, these factors are implicit in the figure too. For instance, trust being socially constructed implies it will be different in other settings. In the SalarySystem project, the Russians had problems understanding the Norwegian tax and salary rules. Consequently, different skills will be associated with competence in other projects.

The relationship between trust and the attributes of trust needs to be evaluated too. However, the purpose of dividing trust into attributes was to make trust identifiable in the GSO context, so the attributes all describe the same phenomenon in a certain sense. Yet, as seen in figure 5.1, the attributes of trust had different degrees at the same time, and hence they describe different aspects of trust. Furthermore, the findings from the SalarySystem project confirmed that predictability and trust are not the same, so the composition of the attributes is important. The issue of what attributes must be present in order to build trust, is not considered to a large extent in this thesis. A study of this issue would be interesting though.

5.2 Implications for practise

In this thesis, figure 5.3 is the suggested model to identify and describe trust in GSO projects. Implicit in this model is the *factors* being addressed by the *attributes of trust*. Also, the model suggests how the attributes of trust influence each other. This way of understanding trust can be used to suggest ways of performing successful GSO projects. In this section, knowledge sharing in the SalarySystem project is applied to illustrate how the model can be used to understand trust.

In the SalarySystem project, knowledge sharing across the sites was problematic. Using figure 5.3 to understand this issue, the problem can be defined as an attempt to build knowledge-based and competence-based trust. Moreover, since the model suggests that knowledge is easier to gain if the appropriate structural mechanisms are in place, such mechanisms could increase the knowledge sharing in the project. In the SalarySystem project, structural mechanisms like TestTool, the test cases, and the improved weekly reports were found to be useful for the increased knowledge sharing in the project. In other GSO projects, mechanisms like contracts, deadlines, milestones, and meetings might be useful in the same way.

According to the knowledge attribute, whose characteristics are implicit in the suggested model of trust, such trust can be built as a result of first-hand experience. Thus, one way to improve knowledge sharing could be to increase the frequency of face-to-face meetings across the sites. Knowledge-based trust also comes from stereotypes, so another way of addressing the problem of knowledge sharing, is to choose the partner from a country being similar to one's own in order to reduce differences in culture and language. Thus, characteristics of the partner being familiar are considered positive for knowledge-based trust to be built.

In the SalarySystem project, the "Cultural Liaison" was useful when building competence-based and knowledge-based trust, and since the attributes of trust influence each other as seen in figure 5.3, predictability and calculation is more likely to appear as a result of his presence.

In the descriptions of the attributes of trust, several problems related to knowledge sharing in GSO projects are identified. The description of predictability suggests that full information about the partner is impossible to get, the description of competence argues that demonstrating competence develops slower in GSO projects than in a face-to-face situation, and, at last, the knowledge attribute recommends bringing together key personnel as a way of building knowledge-based trust. The way the attributes address the problem of knowledge sharing, is an implication for practise too. For instance, the problem of management overhead in GSO-projects is addressed by suggesting face-to-face meetings as a way of sharing knowledge. Thus, such meetings should be taken into consideration when estimating the associated management overhead of a project. Furthermore, since trust is seen to change over time, the overhead related to knowledge sharing will change as the project matures.

In this section, the suggested model of trust has been used to identify and describe trust in GSO relationships, and this understanding was used to suggest ways of performing successful GSO projects.

Chapter 6

Conclusion

In this chapter, the results and contributions of this thesis are summarized. The main objective has been to **identify and describe trust in Global Software Outsourcing relationships**. When discussing this objective, the goal has been to provide a better understanding of trust, and use this understanding to suggest ways of performing successful GSO projects.

Using an interpretive research approach the SalarySystem project, having a Norwegian customer and a Russian supplier, was studied. Based upon the study of this project and relevant literature, six attributes of trust were suggested to identify and describe trust in GSO relationships. The six attributes are:

Predictability: Has to do with observing characteristics about the partner, and expecting these to continue. However, in cases where predictability is possible, the observed characteristics are not necessarily the preferred ones.

Competence: Competence of different kinds, like knowledge about the domain of the product and technical capability, is important in order to perform successful GSO projects. Such skills might be difficult to demonstrate because of geographical distance.

Structure: Structural mechanisms include standards, rules for response time, reporting mechanisms, and written messages. The key function of these structures is to enable predictability.

Calculation: Calculation means assessing whether a relationship will be advantageous which includes both associated risks and opportunities. Such trust is considered important not only in the early stages of GSO relationships, but also in more mature relationships.

Goodwill: Goodwill as a source of trust refers to a general tendency to trust others. Individuals are different with respect to how likely they will develop trust.

Knowledge: Long-time interaction increases the knowledge about the partner. This knowledge makes it easier to predict the future.

In order to have explanatory power in the GSO context, eight factors were considered necessary to be addressed by the attributes of trust. After having applied the attributes of trust on the findings from the SalarySystem project, all factors were addressed by the attributes of trust. These factors find trust to be as follows:

- Multidisciplinary
- Present within *and* between organizations
- Multilevelled
- Related to humans *and* systems
- Changing over time
- Varying in degree
- Bidirectional
- Socially constructed

Based upon the findings from the SalarySystem project, figure 6.1 on the following page is the suggested model to identify and describe trust in GSO projects. The arrows in the figure show how the attributes of trust are seen to influence each other. Implicit in this model is the factors and the attributes of trust.

The suggested model of trust was used to identify and describe trust in GSO relationships. Furthermore, this understanding was used to suggest ways of performing successful GSO projects.

6.1 Future research

During the work on this thesis, several unresolved issues were identified. These will be described below.

Since new projects between ScanSys and RussCo were launched by the end of the SalarySystem project, it could be interesting to further explore this relationship. One objective could be to evaluate how learning from the problems in the SalarySystem project shaped the new projects.

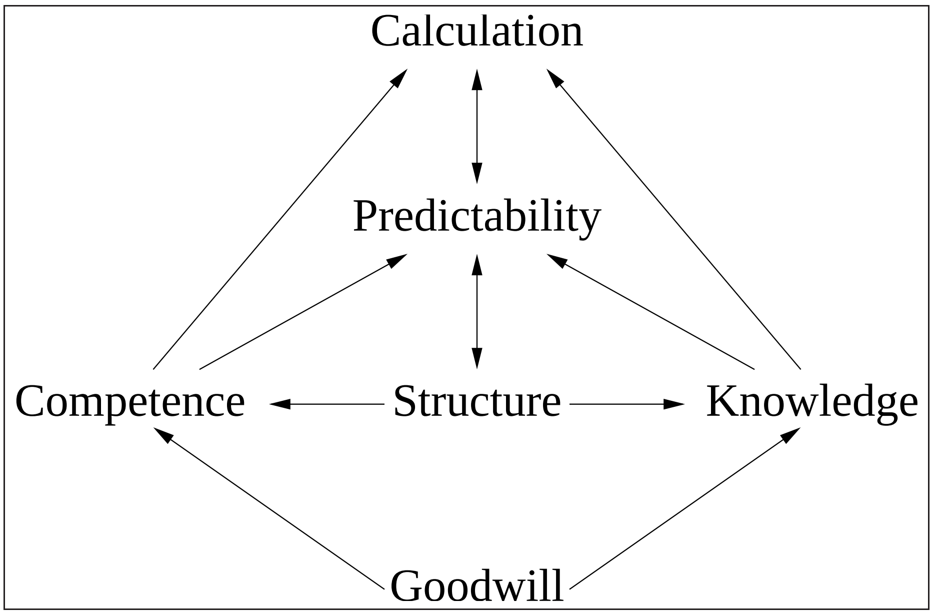


Figure 6.1: The relationships between the attributes of trust.

The results and contributions of this thesis are based upon a single case study. Thus, exploring the suggested model of trust beyond the SalarySystem project would be useful in order to validate it. The model could be applied either to new projects between ScanSys and RussCo, or other GSO relationships.

Several concepts connected to trust were left untouched in this thesis. Although the attributes of trust were used to describe and identify trust, the *composition* of the attributes needs to be further studied. A research question could be: "What attributes are necessary in order to build trust?" In chapter 1; risk, control, power, and culture were seen in relation to trust. In the GSO context, a study of these in addition to trust could be interesting.

References

Beck, U. (2000). *What Is Globalization?* Polity Press.

Braa, K. and R. Vidgen (2000). Research. In K. Braa, C. Sørensen, and B. Dahlbom (Eds.), *Planet Internet*, Chapter 12, pp. 251–276. Lund, Sweden: Studentlitteratur.

Brenkert, G. G. (1998, April). Trust, Morality and International Business. *Business Ethics Quarterly 8*(2), 293–317.

Business Software Association (2002, June). Seventh Annual BSA Global Software Piracy Study. Internet. `http://www.bsa.org/usa/policyres/admin/2002-06-10.130.pdf`. Last read 29th July 2003.

Carmel, E. and R. Agarwal (2001, March/April). Tactical Approaches for Alleviating Distance in Global Software Development. *IEEE Software 18*(2), 22–29.

Carmel, E. (1999, January). *Global Software Teams*. Prentice Hall PTR.

Castelfranchi, C. and R. Falcone (2000, January). Trust is much more than subjective probability: mental components and sources of trust. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Volume 6, pp. 1778–1787.

Child, J. (1998). Trust and International Strategic Alliances: The Case of Sino-Foreign Joint Ventures. In C. Lane and R. Bachman (Eds.), *Trust Within and Between Organizations*, Chapter 9, pp. 241–272. Oxford University Press.

Das, T. K. and B.-S. Teng (2001, February). Trust, Control, and Risk in Strategic Alliances: An Integrated Framework. *Organization Studies 22*(2), 251–283.

Elangovan, A. R. and D. L. Shapiro (1998, July). Betrayal of trust in organizations. *The Academy of Management Review 23*(3), 547–566.

Elmasri, R. and S. B. Navathe (2000). *Fundamentals of Database Systems, 3rd Ed.* Addison-Wesley.

Fujitsu Invia Group (2002, August). Market place – Norway. Internet. `http://invia.fujitsu.com/about/marketplace/norway.shtml`. Last read 29th July 2003.

Gallivan, M. J. and W. Oh (1999). Analyzing IT Outsourcing Relationships as Alliances among Multiple Clients and Vendors. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, pp. 1–15.

Gartner (2003, April). Gartner Says Offshore Outsourcing Market in Europe Will Grow More than 40 Percent In 2003. Internet. `http://www4.gartner.com/5_about/press_releases/pr11apr2003a.jsp`. Last read 29th July 2003.

Giddens, A. (1990). *The Consequences of Modernity*. Stanford University Press.

Giddens, A. (1999, November). Runaway World. The Reith Lectures revisited. BBC. Lecture two about risk.

Grey, C. and C. Garsten (2001, February). Trust, Control and Post-bureaucracy. *Organization Studies 22*(2), 229–250.

Heeks, R., S. Krishna, B. Nicholson, and S. Sahay (2001, March/April). Synching or sinking: Global software outsourcing relationships. *IEEE Software 18*(2), 54–60.

Herbsleb, J. D. and D. Moitra (2001, March/April). Guest Editors' introduction: Global software development. *IEEE Software 18*(2), 16–20.

Hertzum, M. (2002, January). The importance of trust in software engineers' assessment and choice of information sources. *Information and Organization 12*(1), 1–18.

ICPC (2003). Fact sheet. Internet. `http://icpc.baylor.edu/icpc/factsheet.pdf`. Last read 29th July 2003.

IEEE Std 830-1998 (1998). *IEEE Recommended Practice for Software Requirements Specifications*. New York, NY: Institute of Electrical and Electronic Engineers, Inc.

Imsland, V., S. Sahay, and Y. Wartiainen (2003). Key issues in managing a Global Software Outsourcing relationship between a Norwegian and Russian firm: Some Practical Implications. In *Proceedings of IRIS26*, Haikko, Finland. In press. Publication is due autumn 2003.

Jørgensen, M. (2001, August). Prosjektstyring. Lecture in IN219, University of Oslo. `http://www.ifi.uio.no/in219/h01`. Last read 29th July 2003.

Karahannas, M. V. and M. Jones (1999). Interorganizational systems and trust in strategic alliances. In *Proceeding of the 20th international conference on Information Systems*, pp. 346–357. Association for Information Systems.

Khan, N., W. L. Currie, V. Weerakkody, and B. Desai (2003, January). Evaluating Offshore IT Outsourcing in India: Supplier and Customer Scenarios. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pp. 239–248.

Kim, K. and B. Prabhakar (2000). Initial trust, perceived risk, and the adoption of internet banking. In *Proceedings of the twenty first international conference on Information systems*, pp. 537–543. Association for Information Systems.

Klein, H. K. and M. D. Myers (1999, March). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly 23*(1), 67–93.

Kobitzsch, W., D. Rombach, and R. L. Feldmann (2001, March/April). Outsourcing in India. *IEEE Software 18*(2), 78–86.

Kramer, R. M. (1999). Trust and Distrust in Organizations: Emerging Perspectives, Enduring Questions. *Annual Review of Psychology 50*(1), 569–598.

Lane, S. (2003, June). Software Development in Russia. Research report, Aberdeen Group.

Lee, J.-N., M. Q. Huynh, R. C.-W. Kwok, and S.-M. Pi (2003). IT Outsourcing Evolution—: Past, Present, and Future. *Communications of the ACM 46*(5), 84–89.

Maguire, S., N. Phillips, and C. Hardy (2001, February). When 'silence = death', keep talking: Trust, control and the discursive construction of identity in the canadian hiv/aids treatment domain. *Organization Studies 22*(2), 285–310.

Marriot, I. (2003, April). Offshore Sourcing: What Does the Future Hold? In *Outsourcing & IT Services Summit 2003*. `http://www4.gartner.com/2_events/conferences/2003/asm4i/asm4i.jsp`. Last read 29th July 2003.

McFarlan, F. W. and R. L. Nolan (1995). How to Manage an IT Outsourcing Alliance. *Sloan Management Review 36*(2), 9–23.

McHenry, W. and L. Malkov (1999). The Russian's federation's Y2K policy: too little, too late? *Communications of the AIS 2*(2es), 1.

McKnight, D. H. and N. L. Chervany (2001). Trust and distrust definitions: One bite at a time. *Lecture Notes in Computer Science 2246*, 27–54.

McLaughlin, L. (2003, May/June). An Eye on India: Outsourcing Debate Continues. *IEEE Software 20*(3), 114–117.

Nance, K. L. and M. Strohmaier (1994). Ethical Accountability in the Cyberspace. In *Proceedings of the conference on Ethics in the computer age*, pp. 115–118. ACM Press.

Nasscom (2003). Indian Software and Services Exports. Internet. `http://www.nasscom.org/artdisplay.asp?cat_id=314`. Last read 29th July 2003.

Nicholson, B. and S. Sahay (2001, January). Some political and cultural issues in the globalisation of software development: case experience from Britain and India. *Information and Organization 11*(1), 25–43.

NIF (2002). Norske Sivilingeniørers Forenings lønnsstatistikk. Internet. `www.nif.no`. Last read 29th July 2003.

Patrick, A. S. (2002, November/December). Building Trustworthy Software Agents. *Internet Computing, IEEE 6*(6), 46–53.

Pries-Heje, J., R. Baskerville, and G. I. Hansen (2003). Russian High-speed Software Development: Overcoming the Challenges of Globalization. In *Proceedings of the IFIP WG 8.2 Conference*, pp. 253–269.

Repstad, P. (1998). *Mellom nærhet og distanse*. Universitetsforlaget AS.

Reve, T. and E. W. Jakobsen (2001). *Et verdiskapende Norge*. Universitetsforlaget.

Rousseau, D. M., S. B. Sitkin, R. S. Burt, and C. Camerer (1998, July). Not so different after all: A cross-discipline view of trust. *The Academy of Management Review 23*(3), 393–404.

Sabherwal, R. (1999, February). The role of trust in outsourced IS development projects. *Communications of the ACM 42*(2), 80–86.

Sahay, S., S. Krishna, and B. Nicholson (2003). *Global IT Outsourcing: Management of Software Development Projects*. Cambridge: Cambridge University Press. Not yet published—available from August 2003.

Sommerville, I. (2001). *Software Engineering* (6th ed.). Addison-Wesley.

St.Amant, K. (2002, April). When Cultures and Computers Collide: Rethinking Computer-Mediated Communication according to International and Intercultural Communication Expectations.

Journal of Business and Technical Communication 16(2), 196–214.

Sudan, R. (2000, July). The Andhra Pradesh Experience. *Indian Journal of Public Administration XLVI*.

Sydow, J. (1998). Understanding the Constitution of Interorganizational Trust. In C. Lane and R. Bachman (Eds.), *Trust Within and Between Organizations*, Chapter 1, pp. 31–63. Oxford University Press.

Terekhov, A. A. (2001, November/December). The Russian software industry. *IEEE Software 18*(6), 98–101.

The Russia Journal (2002, September). Outsourcers: nothing to fear from piracy. Internet. `http://www.therussiajournal.com/index.htm?obj=27037`. Last read 29th July 2003.

The Russia Journal (2003, June). Russia sees India as competitor, example in world of outsourcing. Internet. `http://www.therussiajournal.com/index.htm?obj=38814`. Last read 29th July 2003.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems 4*(2), 74–81.

Wartiainen, Y. (2003). Kommunikasjon og Outsourcing: Et Casestudium. Master's thesis, Institute of Informatics, University of Oslo.

Wicks, A. C., S. L. Berman, and T. M. Jones (1999, January). The structure of optimal trust: Moral and strategic implications. *The Academy of Management Review 24*(1), 99–116.

Wood, L. E. (1997). Semi-structured interviewing for user-centered design. *Interactions 4*(2), 48–61.

Yang, C. and J.-B. Huang (2000, June). A decision model for IS outsourcing. *International Journal of Information Management 20*(3), 225–239.

Økonomisk Rapport (2003, March). Norge nytt oljesenter? Internet. `http://www.orapp.no/oversikt/Argang_2003/5521/rapport/5715`. Last read 29th July 2003.

Appendix A

Interview guide

General questions asked initially in the research process:

- What is/has been your role in the project?
- What is your education?
- What are the problems in the project?
- Has geographical distance influenced the project? How?
- Have cultural issues influenced the project? How?
- Have language differences influenced the project? How?
- How is your communication towards the other site?
- What communication tools are used?
- What have you learned from the project?
- What could have been done better?
- Why did you decide to outsource?
- How did you estimate the amount of work?
- How many employees does ScanSys/RussCo have?
- What is the product like?

Specific questions asked as the understanding of the project increased:

- How many customers does SalarySystem have?
- How is TestTool being used?

- Has RussCo customized TestTool? How?
- What does the test cases look like?
- Have the Russians initiated any changes in the product? Which?
- Will you outsource future projects to RussCo?
- Would more meetings help? Would more direct contact help the project?
- When problems arose, did you tell ScanSys?
- How much power does the project manager have to change the estimates?
- What is the chain of command to change the estimates?
- How were detailed technical issues about the product explained?
- Did you count lines in order to do the estimation? Why/why not?
- How is the testing done?
- How is the work being reported?
- Why was the estimates exceeded?
- How is the information infrastructure in Russia? How did it influence the project?
- What contractual obligations exist in the project?
- Why are the Finnish board members against outsourcing to Russia?
- During the first meeting in Norway, how were technical details explained?

Introducing theoretical concepts:

- Do you trust the Russians/Norwegians?
- Do you trust their competence?
- What kinds of "local knowledge" exist in the project?

Ad hoc questions (examples):

- What if the redesign had been done internally in ScanSys?
- When the project manager was removed, how did you react?
- Why was the development of SalarySystem divided into two parts?
- Why did not NetMeeting work?