Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2008 Proceedings

European Conference on Information Systems (ECIS)

2008

Customer-Supplier Issues in Software Development

Satya Pasi
Lappeenranta University of Technology, satya.pasi@lut.fi

Kari Smolander
Lappeenranta University of Technology, kari.smolander@lut.fi

Uolevi Nikula
Lappeenranta University of Technology, uolevi.nikula@lut.fi

CUSTOMER-SUPPLIER ISSUES IN SOFTWARE DEVELOPMENT

Satya Jaya Aparna Pasi, Lappeenranta University of Technology, Department of Information Technology, P.O.Box 20, 53851 Lappeenranta, Finland, satya.pasi@lut.fi

Kari Smolander, Lappeenranta University of Technology, Department of Information Technology, P.O.Box 20, 53851 Lappeenranta, Finland, kari.smolander@lut.fi

Uolevi Nikula, Lappeenranta University of Technology, Department of Information Technology, P.O.Box 20, 53851 Lappeenranta, Finland, uolevi.nikula@lut.fi

Abstract

Many studies list the customer-supplier -relationship related issues among the most important reasons for software project failures. Today most of the software development is done outside the organization actually using the software, so the organizational interface of customer and supplier is something that we just need to live with. Since it is not possible, or desirable, to eliminate this interface, we need proper ways to manage the issues introduced by this organizational interface. In this paper we shall look at the customer-supplier interface and issues related to it. The purpose of the paper is to demonstrate common issues and how they can manifest themselves in the software development context. Our study identified six issues related to customer-supplier interface. The issues were related to knowledge transfer, changes during projects, and power relations between customers and suppliers. The paper has implications both for the practice and research. For practitioners the identification and exploration of the issues will provide a solid basis to manage these problems as they come up in practice. For researchers the paper provides an empirically founded description of issues identified in customer-supplier interface.

Keywords: Information systems development, Software development, Customer-Supplier relationship

1 INTRODUCTION

Software development projects today have often an additional task of combining the forces of separate organizations, the customer and supplier organizations. In many cases a working customer and supplier relationship is crucial for a software project success. Since the customer and the supplier are usually separate organizations the issues of the interface between these organizations are often complicated. This scenario can be even more complex in software outsourcing organizations. A number of studies have focused on the success of software projects, and reported that incomplete application domain knowledge, changing requirements, and communication issues are common problems in this area (Alexander et al., 1988; Boehm, 2002; Hanssen et al., 2006; Manhart et al., 2004).

Customer issues are factors that, when present, can adversely affect a project unless project managers take appropriate countermeasures (Wallace and Keil, 2004). The customer has been seen as one of the main software project challenge in some studies (Keil et al., 1998); Wallace and Keil, 2004), and more generally stakeholder related issues have been identified as the main reason for software project failures in more than 8000 projects (Johnson, 1999). Boehm (2002a) has gone so far that he claims that the communication between the customer and the development team is the base for five out of six reasons for software project failure. According to Johnson (1999) the main reason for the cost overruns is problems in customer relationship, and Jorgensen and Molokken-Ostvold (2006) claim in their work that the average cost overruns lie close to 34 % of total development costs. Keil et al (1998) note that even though requirements misunderstanding and change management are critical issues in software project, they can be controlled by the project manager. Customer related issues, however, cannot be controlled by a project manager since they are related to the customer's behaviour (Keil et al., 1998). Therefore, it is of utmost importance for the project manager to forecast the issues that arise from the customer to achieve success in project implementation.

Our previous study explored the context of software development process and as a result provided a list of contextual factors that affected the process (Bern et al., 2007). Out of the 9 factors we realized that the customer is an important factor as it was seen similarly by all the interviews in the collected data. Hence we decided to study the customer on a more detailed level in this study. Although researchers have identified the customer relationship as one of the software management challenges, studies that explicitly investigate customer-supplier related issues are rare. In this study, we analyze customer-supplier issues in the development of tailored systems to help fill this gap. From the set of empirical data collected and analyzed using grounded theory approach, our study reveals a list of customer-related issues in software development. The interview-based empirical data was collected from three software development organizations, where the interviewees ranged from senior and project managers to system architects, analysts, and developers.

The structure of the paper is as follows. Section 2 describes the used research method and studied organizations. Section 3 presents the customer-related issues in software development identified in this study, interleaved with discussion and findings from the literature. Section 4 finally concludes the paper.

2 RESEARCH PROCESS AND METHODOLOGY

The fact that the customer-supplier issues have not received much attention in the software development research before makes the topic well suited for qualitative research approaches. Yin (2003) claims that qualitative research approaches are best suited to areas that lack previous theories, and Benbasat et al. (1987) have found them suitable for in-depth study in a given organizational setting. The grounded theory approach (Glaser and Strauss, 1967; Strauss and Corbin, 1990) is a general qualitative research methodology for the data collection and analysis that uses a systematically

applied set of methods to generate an inductive theory about a substantive data. Seaman (1999) reports that the theory-forming grounded approach suits well for the identification of new theories and concepts. Grounded theory has been found to fit the study of software and information systems (Hansen and Kautz, 2005; Kirsch and Haney, 2006; Seaman, 1999).

The data was collected using theme-based interviews during the beginning of October 2006 and January 2007. The interviews covered four themes: background information; systems development projects methods and practices; effects of the company and its business environment on systems development practices; and networking and cooperation. The interviews were done with upper managers, project managers, and developers (Table 1). Developers in our interviews included system developers, system analysts and system architects. In practice, a single employee may have multiple roles in development and therefore we decided to term these as "developers". The interviews included frequent elaboration and clarification of the meaning, and after all the interviews were audio recorded, they were transcribed to text yielding over 500 pages of transcripts.

Org	Offered products and services	Size (employees)	Interviewees	Duration of the Interviews
A	Operative systems for an industry	800	2 upper managers, 3 project managers, 2 developer; in total 7	9.5 hours
B	Automation systems for one industry sector	100	4 upper managers, 2 project managers, 1 developer; in total 6	5.5 hours
C	Software development services, subcontracting	200	2 upper managers, 1 project manager, 2 developers; in total 5	4.5 hours

Table 1: Summary of the participating companies and interviewees

The data was analyzed using the Atlas.TI (2007) software. The first phase of analysis, *open coding* (Strauss and Corbin, 1990), was conducted by marking any text passage referring to customer. These marked sections were conceptualized and recorded as codes by assigning a label that identified the customer-supplier related issues. Next phase, *axial coding* (Strauss and Corbin, 1990), was to relate the categories to their subcategories to form more precise and complete explanation of the categories. The main purpose of axial coding is to restructure the categories that were found during open coding. We started the axial coding phase by grouping similar codes into broader categories and, for example, *customer influen*ce and *customer participation* were grouped under *customer effects* while *insufficient requirements* and *incomplete knowledge on customer business* were grouped under *insufficient domain knowledge.* At the same time we also learned about the causalities among the categories as, for example, both *insufficient knowledge on customer's business* and *language barrier* had caused *insufficient requirements* as the supplier fails to understand the real customer needs in both the cases. The final phase of integrating and refining the categories to form a theory is called *selective coding* (Strauss and Corbin, 1990). The purpose of this phase is to design a theory on the basis of the categories that were established during the axial coding. First, the theory is integrated by organizing the categories around a central explanatory concept, and then refined by describing poorly defined categories in more depth and excessively described categories are trimmed to form the actual theory; overall Strauss and Corbin (1990) call this phase *theory refining.* In our study we had customer-supplier issues as the central explanatory concept, and all the categories that were found in axial coding were organized around this concept.

3 RESULTS

The analysis of the data was initiated with the goal of identifying the central customer-supplier issues in software development. As the analysis continued, the four key issues started to emerge from the data, and the following detailed research questions were developed to focus the rest of the study:

- Does the supplier have sufficient domain knowledge to develop software?
- How are changes managed during the projects?
- What kind of knowledge transfer difficulties are there between customer and supplier?
- What kind of a relationship do customer and supplier share?

The analyzed data suggested that customer-supplier issues are caused by knowledge transfer problems between the customer and supplier organizations, changes introduced by the customer, and customer dominance over the supplier. Continued analysis of the data revealed five more detailed issues in the customer-supplier interface. However, these issues are more detailed issues of the two earlier identified categories – knowledge transfers difficulties and customer dominance. Thus, in the end of the analysis we had the following three main issue categories with five more detailed issues:

- Knowledge transfer difficulties
  - Insufficient domain knowledge
  - Incompatible terminology
  - Language barrier
- Change management during the project
- Customer dominance
  - Customer dominance on process
  - Business induced power asymmetry

The rest of this section describes each of the identified issue in detail, how they were observed in the analyzed material, and how each issue has been identified and dealt with in related studies. The analysis of the data was done in an inductive fashion, i.e. before in-depth study of the literature, but we added literature references at appropriate places, because we wanted to bring more clarity to the issues that we identified.

3.1 Knowledge transfer difficulties

The three issues in this category deal with knowledge. Knowledge transfer is described as an exchange process of information and skills between entities in a systematic manner (Wanga et al., 2003). Knowledge transfer is done between entities through any forms of communication (Hanssen and Faegri, 2006). Communication plays a vital role in requirements gathering (Holtzblatt and Beyer, 1995). Too much communication will result in depravity of developer's performance (Hanssen and Faegri, 2006), but if there is not enough communication or the communication fails then developers may develop wrong kind of software (Boehm, 2002b).

Knowledge transfer difficulties in the case companies were caused by insufficient domain knowledge, incompatible terminology and language barrier. All three issues cause lack of information to the supplier and may result in misunderstanding requirements and inability to deliver useful software to the customer. The rest of this section describes each knowledge transfer difficulty individually with the observations and consequences.

3.1.1 Insufficient domain knowledge

Insufficient domain knowledge refers to the situation where the customer's application domain knowledge is unobtainable to the supplier. The customer is either not able or willing to deliver application domain knowledge to the suppliers. Rus and Lindvall (2002) define application domain

knowledge as the knowledge about customer's business process, business rules, activities, needs and the customer's business objectives for the software business. Tiwana (2004) illustrates how the application domain knowledge forms the basis for requirements of a software project, and Alexander and Judy (1988) conclude that lack of application domain knowledge can result in inelegant problem-solving strategies. Finally, Curtis et al (1988) claim that software designers often lack knowledge about customer's work and therefore are unable to design useful tools for the customer.

The same issue of limited application domain knowledge was observed in many of our interviews. The limitations in customer side to give further information about their business needs and goals were one of the reasons for having deficient application domain knowledge.

We still understand very little about customers' business in many cases. Reason being restrictions that we sometimes have because of the nature of the customers' business. They won't tell us what their actual goals are.- Senior manager

Actually we do not know business needs which are known to our customer. Why these features are needed in this project is unknown to us. What improves our knowledge is the customer's involvement into the initial preparation of the project, initial preparation of the requirements. If we know the business needs of our customer we can better prepare the software. – Project manager

In this issue the application domain knowledge is unobtainable by the development team which eventually results in customer dissatisfaction as the development team can not address the customer's real need. Even though limited access to domain knowledge as such is a very understandable constraint and can be caused by, for example, confidentiality issues in business or the nature of customer's business, its negative effects on the software project are also undeniable.

According to the literature there are some useful tactics to combat the unobtainable business application knowledge. Keil et al (1998) note that customer should be involved in the project and they even suggest that the project should be led by the customer community. In addition, the customers must be continually reminded of the important role that they play in defining the project's functionality (Holtzblatt and Beyer, 1995; Keil et al., 1998). Further it is shown that project success is correlated with good requirements (Verner et al., 2004) and good requirements can be obtained only from the customer defining them.

3.1.2 Incompatible terminology

Incompatible terminology is quite a common issue in knowledge transfer in software projects. Stiller and Leblanc (2002) have defined incompatible vocabularies as incompatibility between the different vocabularies of software developers and application domain experts, such as users and customers. Since "terminology" is used more in the literature than "vocabularies" (Maidantchik et al., 2002; Nuseibeh and Easterbrook, 2000) in this study the term *incompatible terminology* is used for this issue.

According to Stiller and Leblanc (2002) incompatible terminology creates misunderstandings in the requirements phase and ultimately causes requirements dissatisfaction. Similarly, Van Lamsweerde et al (1998) state that one of the reasons for requirements conflict is the incompatible terminology.

Sometimes we have problems that we don't really understand each other. As we are using different terms, they [customer] don't know the IT terms and sometimes we [supplier] do not understand the business terms. They have very specific terms … and sometimes [the incompatible terminology problem] will disappear when we work together for a long time but often we can still find some new words and concepts. - System architect

We noticed that developers do not have the knowledge of the application domain terms and at the same time the customers do not have all the needed technical knowledge due to their different backgrounds. Incompatible terminology can cause requirements misunderstandings and lead to the development of software that does not meet the customer's expectations.

Literature suggests a few solutions for the incompatible terminology issue. Sommerville and Sawyer (1997) suggest that a supplier team should define a reusable set of terms that can be used in similar projects. Maidantchik et al. (2002) noticed that presenting a terminology proposal that is already accepted by other customers will have a better reception and thus also suggests that an organization should have a predefined terminology proposal. Finally, Stiller and Leblanc (2002) suggests that customers and suppliers with different backgrounds have different understanding of terms, and hence much discussion must occur among the various groups involved in systems development to avoid misunderstandings, errors, and implicit assumptions.

3.1.3 Language barrier

Language barrier refers to the issues that are caused by the limited knowledge in the used language. This difficulty in knowledge transfer and communication in general can cause incomplete knowledge about the requirements and may also lead to wrong requirements as the development team will not understand them.

Chen and Lin (1998) claim that language barrier can be fatal in system analysis and design, especially when developing application oriented products. Welch et al. (2001) state that language-related issues can come up in document translation and the consequences are clearly seen in communication and information flow. They further claim that due to language difficulties parts of vital information can be deleted either deliberately or otherwise. Marschan et al (1997) have similar experiences of vital information deletion and transfer of information as a consequence of language difficulties. Finally, Feely and Harzing (2003) state that "It is worth noting that the language barrier triggers a whole range of negative consequences. It breeds uncertainty and suspicion, accentuates group divides, undermines trust, and leads to polarisation of perspectives, perceptions and cognitions."

The knowledge of English within the customer organizations was reported to be very varied. In some situations the customer was not able to clarify the development team's questions of the requirements because of the language barrier and, consequently, the development team was left without accurate interpretation of the requirements. This was found a common problem in the development of complex systems in a multi-national context.

We always have language problems because our people speak, write and read English more or less well but sometimes we have problems from the customer side. The customer is the one who knows everything about the business area and the product, but sometimes they do not speak English or speak it very badly. This is a problem because it makes it hard to clarify the requirements. Sometimes we ask customers about domain information or requirements but they don't give us answers, and it is a problem. – Systems developer

If we try to describe something in the project and then notice you miss information that you have, for example, in Finnish. Then you try to translate that information to a European language, let's say to German. Or when you try to write in English you notice you do not know how to say that in English, and you just kind of leave it out. - Project manager

From the quote we can see another problem of the language barrier – the loss of required information in documents, for example when the documentation is in a foreign language. It was observed that translation of the documents into a foreign language led to a deletion of a few lines. This happened because the customer team was not able to get the correct words or a meaning for the lines that were in their native language and hence deleted them. The ability to communicate effectively in transferring knowledge is the key to team based success. The deletion of parts of vital information and inefficient communication due to a lack of fluency in language leads to a lack of information resulting in requirements misunderstanding.

A few ways to solve the language barrier issues have been suggested by the literature. For example, Krishna et al (2004) suggest that language training should be given to both the supplier and customer employees. Reel (1999) emphasizes also the importance of good communication that must be built in

all the levels of cooperation between the supplier and customer organizations to avoid problems with knowledge transfer. Further, Welch et al (2001) suggest that a common language could be adopted for example, English can be one solution even though a third language can add difficulties too. Finally, Feeling and Harzing (2003) state that a rational and obvious response to the language barrier is to employ external resources such as translators and interpreters.

3.2 Change Management during the Project

Change management refers to the changes that are introduced during the project by the customers. The requirements that are given in the beginning of the project are often subjects to change which causes a problem as changes require extra resources and time which are seldom given by the customer.

Successful software has to conform to the customer requirements. These requirements are prone to continuous change because of the processes, customers, laws and technology change (Brooks, 1987). Kotonya and Sommerville (1997) stated that "It is often the case that more than 50% of a system's requirements will be modified before it is out into service". Further Manhart and Schneider (2004) claim that customer specific add-ons are always a source of hassle for the developers. Ramzan and Ikram (2005) state that changes that are brought in requirements pose as a problem to developers and project managers. Both Boehm and Ross (1989) and Powell (1996) report technical problems like bugs and low software quality as consequences of such changes. Similar problems were observed also in the present study.

The business needs of our American customers change so much during the projects that it's a real problem to handle all these changes. - Senior manager

Changes in our state law can affect our practices. Probably changes in the USA can affect our work also since law changes in our customer countries cause changes inside the organization or in the business of our customers. - Project manager

Not all customers are ready to expand budget or compromise the schedules when changes come. - Top manager

But now here in this project in which we had at least two changes already and they have delayed the project since we lack available resources, and even if we find them [the resources] the environment is quite complex for a newcomer so it delays… - Project manager

From the above quote we can note that it is often difficult to implement changes because of deadlines, resource problems, and complex development environments. The changes may ultimately result in poor quality and delayed projects if the enforcement of these changes is unplanned and thus needs extra time and resources that are not given by the customer.

Literature includes some suggestions on how to manage changes. Keil et al. (1998) emphasize that one way to manage changes is to educate customer about their impact on financial matters and schedules, and customer should also know what will not be included in the project. Kotonya and Sommerville (1997) emphasize the importance of having a "change control board" to review all change requests as they appear in a project. Sommerville (2001) stresses on having formal processes for managing changes in an organization. Finally, Wiegers (2003) says that all organizations need to have a version control scheme to identify and manage changes efficiently.

3.3 Customer dominance

The next two issues reflect the power relationship that exists between a supplier and a customer. The first issue is the affect of the customer on the process selection and the second one is about business induced power asymmetry between the supplier and the customer.

The customer dominance has received much less attention in the previous research than the issues discussed previously. However, the study of Axtell et al. (1997) revealed that the user-developer relationship was one of the main problem areas in software development process. Heinbokel et al. (1996) report that high user participation and user orientation correlates negatively with team effectiveness and quality of team interaction. They further says that "our results imply that naïve statements suggesting that user centeredness is all positive, need to be modified" (Heinbokel et al., 1996). Finally, Heinbokel et al (1996) stated about the result of Selig (1986) claiming that there is some evidence about certain forms of user participation may lead to project goals being missed more often. Axtell et al. (1997) reports that there is need for further detailed case studies to validate and examine the problems of user participation.

3.3.1 Customer dominance on process

In this study customer dominance on process is defined as a situation where the customer dominates the software development process selection and creates a situation where the supplier is not able to follow a process of its own. In other words, customer forces alignment of development processes on the supplier team and the software development process gets disturbed. This issue has also been identified by Heinbokel et al. (1996) who report that user participating in software development may be limited due to their own behaviour: "one explanation may be that the process of software development is disturbed by user participation."

Sometimes the customers say that we have no time to follow the process, to make the paperwork so we have to develop something very fast. And in this case we follow the customer. - Project manager

We have our own software development process and we are an ICT service company. But in many cases we are following the procedures and processes that our customer uses. We tailor our project to our customer needs and budget. Customer tells us the processes which are specific for them. - Senior manager

Almost in all cases we follow the procedure that the customer has. – Upper manager

From the above quotes we can see that influence from the customer in process selection leads either to a situation where the supplier is unable to follow their own software process or the process is dictated by the customer. The customer has the total control on the project and the process of developing it. Customer practices may delay schedules and since the supplier is forced to do what the customer wants with methods controlled by the customer, the quality of the final product may suffer. A customer may also delay the project with their way of working by not performing their work on time and in schedule. This is quite evident in the following quote:

The customer wanted to work their own way - even though they initially committed to this project and promised they would work our way. First we were supposed to document the business processes and use cases, and move from there little by little to the design of user interfaces and functionality. But only after we had the first versions of the user interfaces done, the customer started doing the specification work - they wanted this part to be moved over there, and that part should work like this - and all this delayed the specification project. So they really had not understood what it meant to operate the way we had suggested. - Systems analyst

The customer does not necessarily realize the importance of working in similar terms with the supplier. The project tends to get delayed if responsibilities are not fulfilled within the schedule. Changes are sometimes introduced by the customer in a phase which has already been completed. By dictating the development process the control lies with the customer and this is may lead to delayed and poor quality projects, because the supplier has to wait for the customer feedback on every step and work with methods and practices that it may not be accustomed to.

3.3.2 Business induced power asymmetry

The other form of customer dominance is business induced power asymmetry with which we refer to the supplier's fear of losing business. We found in many occasions that a project team accepted customer demands as such because they were afraid of losing the customer. Heinbokel et al. (1996) claim that with higher customer orientation project changes are introduced during the later stages of the development process and thus the development process gets disturbed. They further claim that customer orientation can affect the supplier performance: "customer orientation may lead to high levels of aspiration and, therefore, to a higher degree of stressors and a lower degree of team effectiveness" (Heinbokel et al., 1996). It was also observed in our case companies that accepting customer demands can cause delays and low quality outcomes.

This sale case was active in the fall of 2004, and we did not have too much work at that time... Such a big and attempting project [from a customer] we always wanted to have, and then we got such an opportunity. You would have done anything to get the deal. Still, it should not have affected the work effort estimation and function point calculations - and I trust it did not - but I assume the price was compromised due to the fact that the customer wanted the system to be ready by the end of the year 2006. Our suggestion was half a year later, but when the customer pressured to get it ready by the end of the year, it was assumed that we could probably do it. – Senior project manager

A customer watches the project very closely and coming to look what we are doing he might say a small thing like "Change this" and probably we do it. - Systems analyst

It is very important for us to maintain the discipline in the company.. .that the customer is the God and the person who you work for and ... that you have to solve your problems yourselves. –Upper manager

We can say that the relationship between the supplier and the customer is asymmetrical, because the customer has usually a higher power over development projects than the supplier. The supplier has to adjust to the demands of the customer that may result to change requests at the final stages of the development. We observed that a supplier cannot often say no to the customer change requests that are introduced in the projects. This leads to changes in the software requirements at a very late stage of the project and may result to delayed schedules or poor quality software projects. Unfortunately, the literature does not yet provide suggestions on how to address this problem.

4 CONCLUSION

The importance of empirical studies is to authenticate the relative value of results within a field of study which will contribute to the advancement of that field (Seaman, 1999; Wasson, 2004). In this study the data collected from the companies helped us to understand how customer issues often arise in the requirements engineering and communication domains. Based on a set of in-depth interviews this study describes three customer-supplier related issues in software development.

As shown in the results section, our results regarding the understanding of requirements engineering and communication between the customer and developer have been similar to existing studies (Boehm, 2002a; Hanssen and Faegri, 2006; Johnson, 1999; Keil and Carmel, 1995; Keil et al., 1998). This study has, however, brought a unified view to the customer-supplier related issues and added customer dominance to the list of issues.

We found that unobtainable knowledge of customer's business and knowledge transfer difficulties lead to insufficient requirements and lack information on customer's business needs. Changes introduced during the project results in poor quality and delays in the project. Customer dominance also introduces changes as the development team often cannot refuse the customer's requests. By influencing the way software is developed, the customer is dominating and dictating the development processes and it is questionable how beneficial this is to the customer. Table 2 shows the summary of the observed issues with their effects.

This study explained how customer-related issues are frequently happening in software development practice. The study has its limitations. All the studied case organizations were developing tailored software and therefore similar kind of observations can be expected primarily in tailored development. Product-oriented development may have its specific issues, but it is possible that many of the issues mentioned here are common. Another major limitation is that this study has the supplier viewpoint only. Though we tried to get the customer viewpoint it was not possible due to the restrictions that we had from the supplier organizations who were the participants in the project.

Observed issue		**Explanation**	**Observed effect**
Knowledge transfer difficulties	Insufficient domain knowledge	Customer restrictions to give application domain knowledge to the development team.	Lack of business application domain knowledge leads to requirements misunderstanding and inability to address the customer needs.
	Incompatible terminology	Terms used between developer and customer are often difficult to understand. Customer does not understand IT terms and developers do not understand application domain terms.	Terms are understood differently which leads to requirements misunderstanding and inability to develop useful software for the customer.
	Language barrier	Difficulty in speaking with each other and translating documents because of not having fluency in spoken and written language.	Development team lacks information needed from the customer which leads to requirements misunderstanding.
Changes brought during the projects		Customer introduces changes in the project as laws, technology and business needs change.	Poor quality and delayed projects.
Customer dominance	Customer dominance on process	Customer is dictating the software development process. Customer does not work along with the development team and thus the schedules of customer and development team differ.	Project is developed haphazardly as it lacks the development team's control on the software development process leading to poor quality software and delays in project schedule.
	Business induced power asymmetry	Development team does not refuse customer requests due to the fear of loosing business.	Constantly responding to customer demands leads to poor quality software and schedule delays.

Table 2: Observed issues and their effects.

The practical implication of this research is significant for project managers of tailored software projects. The results enable project managers to forecast the customer-related issues in projects. By knowing such issues, one can be better prepared for their appearance and create strategies for overcoming them. For the researchers our study provides further evidence of the importance of customer-supplier relationship. In particular our data provides evidence of the negative effect a customer can have on a software project through dominance. These indications are still initial, and thus further work is needed. Future study can concentrate on comparing these issues in other kinds of software organizations. To saturate fully these findings, one can include observations for example from a subcontractor, a software product development company, and an end-user software developer organization.

References

(2007) Atlas.TI Scientific Software, Berlin.

Alexander, P. A. and Judy, J. E. (1988) *The Interaction of Domain-Specific and Strategic Knowledge in Academic Performance* Review of Educational Research, 58 (4), pp. 375-404.

Axtell, C. M., Waterson, P. E. and Clegg, C. W. (1997) Problems integrating user participation into software development, International Journal of Human-Computer Studies, 47 (2), pp. 323-345.

Benbasat, I., Goldstein, D. K. and Mead, M. (1987) The case study research strategy in studies of information systems, MIS Quarterly, 11 (3), pp. 369-386.

Bern, A., Pasi, A. S. J., Nikula, U. and Smolander, K. (2007) Contextual Factors Affecting the Software Development Process – An Initial View, 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, Gdansk, Poland, June 5.

Boehm, B. (2002a) Get ready for agile methods, with care, Computer, 35 (1), pp. 64-69.

Boehm, B. (2002b) Software engineering is a value-based contact sport, IEEE Software, 19 (5), pp. 95-96.

Boehm, B. W. and Ross, R. (1989) Theory-W software project management principles and examples, IEEE Transactions on Software Engineering, 15 (7), pp. 902-916.

Brooks, F. P., Jr. (1987) No Silver Bullet Essence and Accidents of Software Engineering, Computer, 20 (4), pp. 10-19.

Chen, Q. and Lin, B. (1998) Global outsourcing and its managerial implications, Human Systems Management, 17 (2), pp. 109-114.

Curtis, B., Krasner, H. and Iscoe, N. (1988) A field study of the software design process for large systems, Communication of the ACM, 31 (11), pp. 1268-1287.

Feely, A. J. and Harzing, A.-W. (2003) Language Management in Multinational Companies, Cross Cultural Management: An International Journal 10 (2), pp. 37-52.

Glaser, B. G. and Strauss, A. L. (1967) The discovery of grounded theory: strategies for qualitative research Aldine De Gruyter, Chicago.

Hansen, B. H. and Kautz, K. (2005) Grounded Theory Applied - Studying Information Systems Development Methodologies in Practice, Proceedings of the 38th Annual Hawaii International Conference on System Sciences HICSS '05.

Hanssen, G. K. and Faegri, T. E. (2006) Agile Customer Engagement: a Longitudinal Qualitative Case Study, Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering ISESE'06, ACM Press, Rio de Janeiro, Brazil, pp. 164-173.

Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W. and Brodbeck, F. C. (1996) Don't underestimate the problems of user centredness in software development projects - there are many!, Behaviour and Information Technology, 15 (4), pp. 226-236.

Holtzblatt, K. and Beyer, H. R. (1995) Requirements Engineering: the human factor, Communication of the ACM, 38 (5), pp. 31-32.

Johnson, J. (1999) Turning Chaos into Success, Software Magazine.

Jorgensen, M. and Molokken-Ostvold, K. (2006) How large are software cost overruns? A review of the 1994 CHAOS report, Information and Software Technology, 48 (4), pp. 297-301.

Keil, M. and Carmel, E. (1995) Customer-Developer Links in Software Development, Communication of the ACM, 38 (5), pp. 33-44.

Keil, M., Cule, P. E., Lyytinen, K. and Schmidt, R. C. (1998) A Framework Identifying Software Project Risks, Communications of the ACM, 41 (11), pp. 76-83.

Kirsch, L. J. and Haney, M. H. (2006) Requirements determination for common systems: turning a global vision into a local reality, The Journal of Strategic Information Systems, 15 (2), pp. 79-104.

Kotonya, G. and Sommerville, I. (1997) Requirements Engineering: Processes and Techniques, John Willey and Sons Ltd.

Krishna, S., Sahay, S. and Walsham, G. (2004) Managing cross-cultural issues in global software outsourcing, Communication of the ACM, 47 (4), pp. 62-66.

Maidantchik, C., Montoni, M. and Santos, G. (2002) Learning organizational knowledge: an evolutionary proposal for requirements engineering, Proceedings of the 14th international conference on Software engineering and knowledge engineering, ACM, Ischia, Italy, pp. 151-157.
Manhart, P. and Schneider, K. (2004) Breaking the ice for agile development of embedded software: an industry experience report, Proceedings of 26th International Conference on Software Engineering ICSE 2004., pp. 378-386.
Marschan, R., Welch, D. and Welch, L. (1997) Language: The forgotten factor in multinational management, European Management Journal, 15 (5), pp. 591-598.
Nuseibeh, B. and Easterbrook, S. (2000) Requirements engineering: a roadmap, Proceedings of the Conference on The Future of Software Engineering, ACM, Limerick, Ireland, pp. 35-46.
Powell, A. L. (1996) A literature review on the quantification of software change, Department of Computer Science, University of York, Heslington.
Ramzan, S. and Ikram, N. (2005) Making Decision in Requirement Change Management, First International Conference on Information and Communication Technologies ICICT 2005, pp. 309-312.
Reel, J. S. (1999) Critical success factors in software projects, IEEE Software, 16 (3), pp. 18-23.
Rus, I. and Lindvall, M. (2002) Knowledge management in software engineering, IEEE Software, 19 (3), pp. 26-38.
Seaman, C. B. (1999) Qualitative methods in empirical studies of software engineering, IEEE Transactions on Software Engineering, 25 (4), pp. 557-572.
Selig, J. (1986) EDV-Management. Eine empirische untersuchung der Entwicklung von Anwendungssystemen in deutschen unternehmen, Springer, Berlin.
Sommerville, I. (2001) Software Engineering, Addison-Wesley an imprint of Pearson Education.
Sommerville, I. and Sawyer, P. (1997) Viewpoints: principles, problems and a practical approach to requirements engineering, Annals of Software Engineering, 3, pp. 101-130.
Stiller, E. and Leblanc, C. (2002) Project-based software engineering, Addison-Wesley.
Strauss, A. L. and Corbin, J. (1990) Basics of Qualitative Research: Grounded Theory Procedures and Applications, Sage Publications, Newbury Park, CA.
Tiwana, A. (2004) An empirical study of the effect of knowledge integration on software development performance, Information and Software Technology, 46 (13), pp. 899-906.
Wallace, L. and Keil, M. (2004) Software Project Risks and their Effect on Outcomes, Communications of the ACM, 47 (4), pp. 68-73.
Van Lamsweerde, A., Darimont, R. and Letier, E. (1998) Managing conflicts in goal-driven requirements engineering, IEEE Transactions on Software Engineering, 24 (11), pp. 908-926.
Wanga, P., Tongb, T. W. and Koh, C. P. (2003) An integrated model of knowledge transfer from MNC parent to China subsidiary, Journal of World Business, 39, 168–182.
Wasson, K. S. (2004) Requirements metrics: Scaling up, Proceedings of 2nd International Workshop on Comparative Evaluation in Requirements Engineering (CERE '04), Kyoto, Japan, pp. 51-55.
Welch, D. E., Welch, L. S. and Marschan-Piekkari, R. (2001) The Persistent Impact of Language on Global Operations, Prometheus, 19 (3), pp. 193-209.
Verner, J., Cox, K., Bleistein, S. and Cerpa, N. (2004) Requirements Engineering and Software Project Success: An Industrial Survey in Australia and the U.S., Australia Journal of Information Systems, 13 (1), pp. 225-238.
Wiegers, K. E. (2003) Software Requirements, Microsoft Press.
Yin, R. K. (2003) Case study research: design and methods, Sage Publications.