Dilemma Structures between Contracting Parties in Software Development Projects

Cornelia Gaebert
Research Group on Strategic Information Management, European Research Center for Information Systems
c/o University of Muenster, Leonardo Campus 11, D-48149 Muenster, Germany

Keywords: Software Development Project, Outsourcing, Failure, Information Asymmetry, Dilemma Structures, Incomplete Contract.

Abstract: The risk of failure of software development projects has been high for decades. One main reason identified by empirical studies is that the requirement specifications have gaps at the start of the project. Research on requirement analysis and project management primarily focuses on the improvement of methods and the behavior of the project participants. In our study, we suggest switching to the contracting level, describing the interaction of the involved organizations in terms of game theory. Organizations pursue economic targets. As we will show, the customer and the supplier are in a dilemma situation regarding the effort for closing the gaps in the requirement specifications. This results in a high risk for the quality of the software system. In support of our theoretical argument, we carried out an empirical investigation that shows that gaps in requirements and conflicts in the project exist of nearly every project. The most commonly used contract model is a fixed-price contract. From our model, we can derive suggestions for the contract design of software development projects as well as for the cooperation behavior during the project.

1 INTRODUCTION

Despite project management improvements and professionalization of the software development process, the number of failing software development projects has remained high for decades (Standish Group, 2010; El Emam and Koru, 2008).

Organizations expect to mitigate this risk by outsourcing (Chua et al., 2012). They expect, that the supplier takes the risk for the project failing when working under autonomy. The customer considers the supplier to be responsible for budget, time, and quality.

Researchers in the field of software project management and software engineering have focused their studies on the project's internal problems, even when external suppliers carry out the projects (Natovich, 2003; Al-Ahmad et al., 2009). Moreover, they provide recommendations for practical action straight from the success factors derived from reasons of failure (see also Dwivedi et al., 2013). They consider the qualifications of all stakeholders as well as the continuous improvement of project management (Buhl and Meier, 2010), such as the change from structured to agile project management (Zannier and Maurer, 2007).

This paper shifts the spotlight on the relationship between customer and supplier. We argue for describing the software development project at the contracting level, as a cooperation of two parties: the one that needs a software system to meet their individual requirements, and the one that has the ability to produce the software system. We call the first party the *customer*, and the second the *supplier*. First of all, both parties pursue economic targets.

The aim of this paper is to show that a formal description of the cooperation between the supplier and customer of a software development project will open new perspectives for understanding the failure of these projects. We provide a theoretical rationale for the failure of software development projects. Therefore, we introduce and justify a model of the software development project as a two-party interaction game, in which the delivery of information is the crucial element in each interaction. Using this approach, it will be possible to analyze contractual situations for software projects with respect to risks of failure. We will show how the structure of this interaction results in a

Gaebert C..
Dilemma Structures between Contracting Parties in Software Development Projects.
DOI: 10.5220/0004996405390548
In *Proceedings of the 9th International Conference on Software Engineering and Applications* (ICSOFT-EA-2014), pages 539-548
ISBN: 978-989-758-036-9

high risk of failure for such projects. Nonetheless, from our model we can derive some suggestions for the contract design of software development projects, and for the cooperation behavior during the project.

We base our argument on the number one reason for failure, incomplete, ambiguous, and changing requirements (Standish Group, 1995; El Emam and Koru, 2008; Standish Group, 2010; Liu et al., 2011; McGee and Greer, 2012). In this paper, we call this deficit *requirement gaps*. As we will show in section 2.1, this is inherent in the setting of a software development project. Therefore, the customer and supplier need to interact with each other to establish clear requirements, explain changes and exchange information over time. In section 2.2 we briefly show the possible behavior of the actors in this situation. Regarding the delivery of needed information, the parties are in a situation called the *prisoner's dilemma*. Therefore, we introduce in section 2.3 the prisoner's dilemma as a formal description (Trucker, 1950). Section 2.4 describes the software development projects in terms of game theory. We must ultimately expect that when both parties defect from cooperation, the project tends to fail. Finally, we show in section 2.5, that even under the mostly agreeable fixed-price contract (Oestreich, 2006), the cusomer want to save costs. However, the customer will be dissatisfied with the quality of the developed software. Consequently, the contract is unable to fulfill its function and the project tends to fail.

In section 3 we support this theoretical argumentation using a two-step empirical investigation. First, we interviewed experts, both customers and suppliers, using a formal questionnaire. Second, we conducted in-depth expert interviews. The empirical results show the relevance of these concepts for the understanding of problems in software development projects.

Finally, in section 4 we summarize these suggestions and describe some directions for further research, starting with this model.

2 THE PROJECT AS A TWO PARTY GAME

Researchers in the field of software project management focus primarily on the control of decisions and activities of the acting participants and stakeholders within the development organization (Keil et al., 2004; Rustagi et al., 2008). They often describe them as rational agents having goals and making decisions for the cooperation with other actors, with the purpose of achieving a maximum of benefit (Yilmaz and O'Connor, 2012; Cockburn, 2004). However, as shown by Tollefsen (2002), we can also consider organizations like companies or public authorities as rational agents who have their own goals and make rational decisions for reaching these goals.

At the organizational level, regarding a software development project, we can define two kinds of actors: First, there are organizations acting as the customer; and second are the organizations acting as the supplier. The customer has business goals that result in requirements for a software system, which are described in a requirement specification document. The supplier has the ability to develop an information system that meets those requirements. Therefore, the customer and supplier sign a contract to carry out a software development project.

2.1 System-Inherent Causes for Incomplete Requirement Specifications

In an ideal world, the requirement specification is complete, unambiguous, and clear. In such a world, the supplier has calculated all efforts for the implementation of the requirements before signing the contract. Based on the specification, the designers and developers will implement the needed system. No communication and no interaction between the parties will be necessary during the project.

Unfortunately, requirements are not complete and unambiguous. As shown in research literature (Liu et al., 2011; McGee and Greer, 2012), and as stated by all experts in our empirical survey (see section 3), gaps exist in the requirements specifications. Researchers and practitioners have exerted a lot of effort in developing methods for producing better specifications without gaps, misunderstandings, and unclear descriptions.

Nevertheless, as we will argue in the following, there are system-inherent causes for the gaps in requirement specifications.

First, software requirement specifications contain knowledge in a strict sense only about the past and the present. For instance, the customer knows problems that exist with the currently used system, the present market situation, and business cases. About the future, there are only assumptions. In particular, how the new system will change the

business processes is not a matter of fact, but a matter of expectation and anticipation.

Second, the requirement engineer can only document consciously available knowledge, and to some extent subconsciously available knowledge. However, in all business processes, relevant conditions and information exist that no one knows about (Kano et al., 1984). The customer has knowledge primarily regarding the business for which the software system is needed. In contrast, the supplier has knowledge regarding technical issues, like the properties of used frameworks and development techniques. Furthermore, on the supplier's side, experiences from other projects regarding user acceptance and performance problems exist. This knowledge is also relevant for the development of a software system, but in the moment of documenting the requirements it is not avaialable.

Third, the software developmet project needs time. The cutomer and the supplier interact and exchange information during the project's development. As their settings change, new requirements may arise.

Consequently, we have to accept the fact, that requirement specifications will contain gaps also in the future, and even if research in requirement engineering finds new and better methods.

2.2 Possible Choices of Rational Actors

As we have shown, the customer and the supplier must sign the contract based on an incomplete requirement specification. Closing the gaps is part of each software development project, and there is no way to avoid this situation. The question arises, how a rational actor will behave in this situation.

Both actors have the choice to participate in the closing of specification gaps, or to avoid these efforts and to demand this effort from the other party. Therefore, we have to analyze four cases.

(1) The customer tries to avoid effort, whereas the supplier exerts effort in closing the gaps. The customer may argue that the supplier should calculate these efforts during the calculation of the projects costs. Furthermore, the supplier has seen the specification before signing the contract and has commited to implement the needed system, if necessary by detailing the requirements. In such cases, customers will argue that there are no real gaps in the requirements but there are some details left to be defined during the system design phase. Thus, the supplier is responsible for specifying these details. The customer will avoid delivering resources for clarification. The supplier must specify assumptions and define suggestions, and the customer is free to accept or to reject them.

The result is an enormous effort on the supplier's side, whereas the customer will save on costs and will get the needed system with little effort on their own part.

(2) The reverse situation is also possible. The supplier can avoid exerting effort in closing the gaps and can demand all information needed from the customer. If the supplier finds a specification gap during the design and the implementation of the system, he will ask the customer for clarification and deny sending their own experts or making his own suggestions based on experiences from other projects. In this case, there will be high costs on the customer's side, whereas the supplier incurs no extra costs for closing the gaps. Furthermore, the supplier has the ability to initiate change requests to get extra payments.

(3) It is possible, that both parties avoid any effort in closing the requirement gaps. The supplier may implement the system without asking the customer if there is a problem with the specification. Alternatively, if the supplier asks, he can be satisfied with any answer from the customer and does not reflect it on own expieriences. The customer may also avoid effort for clarification. Both sides may see the other side as being responsible for closing the gaps and may ignore arising problems. The result of this behavior is that both sides save efforts during the project, but in the end, the system does not meet the real business requirements of the customer. The project is highly risky, and if it fails, the customer will not pay the price for the development. Therefore, in such a case, both parties will probably lose their investments.

(4) Finally, both parties may cooperate, sending their experts and delivering all information and experiences for finding the right solution in the case of requirement gaps. The efforts on both sides then are high; however, the project can finish with a system that meets the requirements.

Clearly, the fourth case is the best way to finish a project successfully. However, in reality, both the customer and the supplier have to save costs by avoiding extra effort. Therefore, it is not self-evident that the parties cooperate as described in this scenario.

For some time *game theory* has described the structure of the situation as *prisoner's dilemma* (Tucker, 1950). In recent years, the prisoner's dilemma has already been used in the analysis of dilemma structures between developers within

software development projects (Hazzan and Dubinsky, 2005; Yilmaz et al., 2010). We will use this model as an analytical tool for understanding the situation of the projects' parties. First, we will introduce the original picture, giving the model its name. Then we will apply it to the project situation.

2.3 The Prisoner's Dilemma

In the prisoner's dilemma, a prosecutor questions two prisoners individually. Both prisoners (player) can deny the alleged offense (cooperate with each other), and both result in an imprisonment of 5 years. However, each of them can also admit and incriminate the other (defect). If only one of them admits, he or she gets the acquittal (leniency) and the other gets 20 years of imprisonment. If both confess, each receives 10 years of imprisonment. Although it would be best for both prisoners, if they denied the offense, they will both confess because of the incentive conditions of the situation. The special situation in capturing the dilemma situation is that both actors miss the potential gains from cooperation just because they follow their own incentives and thus act rationally.

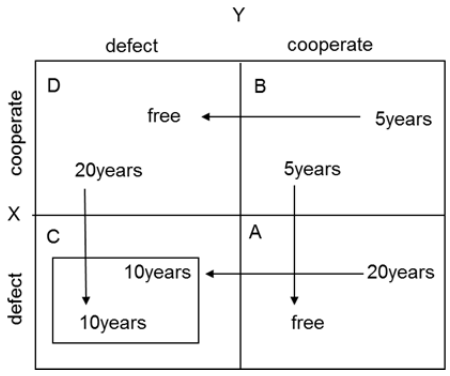

Figure 1: Prisoner's dilemma.

Figure 1 depicts the situation and the preferences of the prisoners in a schematic way. We enter the payoff for each player in four quadrants: A, B, C, and D. We enter the results of player X in the lower-left corner of each quadrant, and we list the payoffs of player Y in the upper-right corner. The arrows in the figure mark the advantage calculi. The horizontal arrows describe the tendency of Y; the vertical arrows describe the tendency of X.

For X and Y, defecting is the dominant strategy, which they will choose. Both prisoners make their rational decisions independently from the other, knowing the possible choices that the other may make. If the other cooperates, for each prisoner it would be best to defect, because he will be free. If the other one defects, for each it is also the better choice to defect. Consequently, both prisoners will defect and will get a bad result. If both decide to cooperate, the result would be much better.

The frame in the lower-left quadrant C shows the (Nash) equilibrium, the result that rational actors will get in a prisoner's dilemma situation.

2.4 the Customer and the Supplier in a Dilemma Situation

Now, for applying the prisoner's dilemma to the software development project, we identify the player Y with the customer and player X with the supplier. If the customer defects and the supplier cooperate, the latter will close all specification gaps at his own cost, and the customer will get the best result (case 1 in section 2.2, quadrant D in Figure 1). In contrast, if the supplier defects by avoiding the needed effort, and the customer works hard to close all gaps, it will be the best for the supplier (case 2, quadrant A in Figure 1). If both parties cooperate, both incur some costs, but they get the best system as the result of the project (case 4, quadrant B in Figure 1). Finally, if neither the customer nor the supplier work on closing the gaps, they will exert less effort, but the result is a bad system that does not meet the requirements (case 3, quadrant C in Figure 1).

If both the supplier and the customer in a software development project act as rational actors, they both must avoid any effort in closing gaps in the requirement specification – the result will be a bad system.

Please note, only the order of evaluation is in this situation crucial for the result, not the concrete rating level (Axelrod, 2009). Therefore, we can translate the payoff to simple numerical amounts for the better representation of the problem structure of the dilemma situation in the form of a prisoner's dilemma (Beckmann and Pies, 2006). Figure 2 depicts the four cases in four fields.

The supplier (player X) gets in quadrant A a result of 4 (only the customer closes gaps). In quadrant B, both get a payoff of 3 (both close gaps), and in C a payoff of 2 (nobody closes gaps). In the D quadrant, the supplier realized his worst result of 1 (only the supplier closes gaps). The customer (player Y) obtains in quadrant D his best result with the payment of 4. The customer achieves his worst result in quadrant A with a payoff of 1. Divergent preferences determine the order of evaluation of possible results: For the supplier, it is A> B> C > D and for the customer it is D> B> C > A. The payoff

matrix of the one is therefore the transposed payoff matrix of the other.

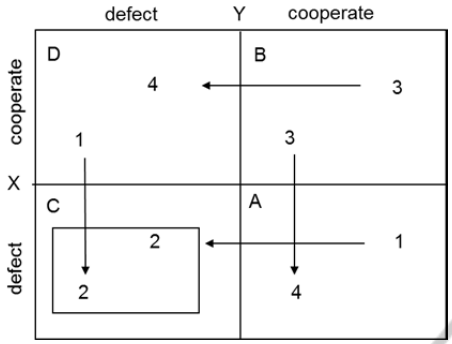

Figure 2: The prisoner's dilemma of the software development project in normal form with payoffs.

The rational actors achieve the dominant result because there is no effective behavior binding, i.e. the supplier and the customer are in a so-called institutional vacuum (Beckmann and Pies, 2006). If the supplier and the customer want to escape this dilemma, they must prevent the institutional vacuum so that they are no longer in a dilemma structure. They can achieve this only through collective self-commitment to cooperation, through simultaneous abandonment of the solutions in the quadrants A and D. Both can improve their payoff only in this way. They must find rules that reward cooperation and punish defection to guarantee effective behavior binding. Following the cooperation agreement must be the rational choice for the actors. Each actor will decide this way, only if the achieved result is better for him than the solution without agreement. The agreement must eliminate the conflict. It causes the actors no longer to operate independently (Gauthier, 1985). The players cooperate only if they know the alternative solutions and if they are sure how the other one will act (Davis, 1985).

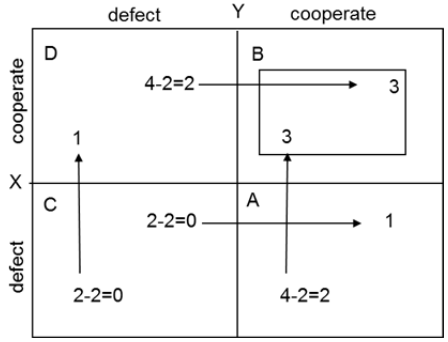

Figure 3: Negative sanction for both.

If a negative sanction is established for both players in the case of defection, the possible payoffs change (Figure 3). The preferences are changing, and so the order of evaluation of the results changes. Cooperation will be the dominant strategy. The enticing thing about this situation is that no actor cares how the other player is set. The individual gets, in any case, a payoff of 3 if he cooperates. The actors found a new opportunity space by way of rules. These rules change the incentives so that the actors can still defect, but they do not want to defect. It is not about improving the game, but about playing another game.

2.5 Rational Behavior in Dilemma Situations

As empirical studies show, contractual arrangements between both parties vary between fixed-price and time-and-material contracts (Kalnins and Mayer, 2004; Fink et al., 2013). Fixed-price contracts consist as the name suggests of a fixed-price for the developed software. In the case of a time-and-material contract, the customer pays for a specified amount per hour. Sometimes, the two contract types are combined, such as a fixed-price for the initial development and time-and-material for its enhancement.

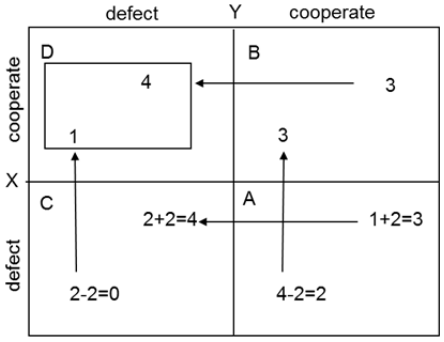

Figure 4: The software development project under fixed-price with one-sided sanctions.

With a time-and-material contract, neither the customer nor the supplier has incentives to avoid efforts in closing the requirement gaps. This contract type eliminates this conflict as long as the customer is willing to pay. However, fixed-price contracts dominate the contract types. Under the conditions of a fixed-price contract, the customer in particular has the chance to exert pressure on the supplier by threatening to reject the system and to deny paying the agreed-upon price. Some authors explicitly

demand to control the suppliers' work in detail (Rustagi et al., 2008).

Because of the sanction for the supplier in such a contract situation, the order of evaluation for the supplier changes (Figure 4). The preferences of the supplier switch from defecting to cooperating. The customer receives the penalty as a bonus, so his preference does not change. His payoff for defection is always higher than his cooperation payoff. The balance adjusts itself in quadrant D, where the customer achieves his best result. The inescapable conclusion of this finding is that the customer will not want to cooperate in closing the gaps because, no matter how the supplier chooses, he always achieves his best result with defection. He just needs to ensure that he collects the penalty from the supplier in the case that the supplier has not demonstrably fulfilled the contract. The actors will not achieve the equilibrium solution in quadrant B.

Nevertheless, the supplier has options to deal with the situation, and he must do this, if he is a rational actor. As shown by Spremann (1990), in the case of asymmetrically distributed information, there are options for hidden actions. In software projects, the supplier has the chance to save effort on quality issues as performance, maintainability, reliability, and other quality attributes. Problems from this behavior will appear after finishing the project, and due to the fact, that there are many possible causes for problems, the supplier may deny the responsibility for these problems. Therefore, also the customer should have an interest in finding a contract design as described in Figure 3.

Is it possible in a software development project as under investigation of this study to implement negative sanctions in the case of defection for both parties? It is not difficult to implement sanctions regarding the supplier. If the supplier does not meet the milestones, or if the quality of the software system is bad, it is possible for the customer to deduct a penalty from the agreed price.

On the other side, a sanction for the customer would mean that he has to pay a higher price. This would escape the fixed-price condition, so it does not seem possible to implement such a sanction.

3 EMPIRICAL SUPPORT OF THE THEORETICAL ARGUMENT

We support our theoretical findings with an empirical survey. First, it is essential that the supplier gets mostly a fixed-price for the software system. If the customer would pay an effort-based price for all of the work done by the supplier, no dilemma situation would arise. Second, do the customer and supplier agree that there are gaps in the requirement specifications delivered by the customer by signing the contract? Third, is there a potential conflict resulting from this situation? Do both parties quite agree that there is conflict? To support the practical relevance of these assumptions, we carried out an empirical investigation.

For this empirical part of our study, we conducted a two-step evaluation. First, we developed a questionnaire in the form of a standardized online survey as a special kind of standardized survey (Klammer, 2005). Next, we conducted personal interviews to deepen our understanding of the results from the questionnaire. The period of the evaluation was one year.

For the questionnaire, we chose the standardized online survey to give the respondents an opportunity to reflect and to question their own companies (Schnell et al., 2011). The format of the online survey itself was legitimate because the interviewees were an IT-savvy group. Open answers supplemented the closed questions to not be too restrictive and to gather the covered information (Mayer, 2012). In the following, we will analyze and interpret the results descriptively.

Experienced project participants on both sides (customer and supplier) were interviewed. The questionnaire had to take the management perspective as well as the view of the project management into account. Because it is not possible to address trivially the population of all manufacturers and customers of custom software, and because questioning the population about any associated unacceptably high cost is not realistic, we chose a smaller population. Therefore, we could not achieve complete representativeness (Schnell et al., 2011). For practical reasons, we addressed the 45 members of a network of IT companies in Germany. Fifty additional addressees were available from other contacts. To expand the circle of respondents and to amplify the customer side, we used contacts in social networks such as Facebook (approximately 30), Xing (approximately 20), and Twitter (approximately 50). This ensured that the respondents had experience in different contexts of possible projects. Of the 200 addressees who were requested to participate in the survey, 29 actually completed the questionnaire (14 suppliers, 5 customers, 9 suppliers and customers (both), and 1 other).

An independent survey that evaluated the willingness to participate in the survey suggested a conscientious answering of the questions. A total of 48.3% of the respondents indicated that they belong to management and that they have responsibility for the contracts; 27.6% are project managers; 6.9% are employees at the working level; and 17.2% perform other activities, such as consulting. A total of 89.7% of the respondents had 10 or more years of experience with software development projects. The participants represented a broad range of sizes of projects with regard to the duration and number of employees.

For the exemplary and in-depth interviews, we conducted semi-structured expert interviews. We questioned, on the one side, a consultant with experience in software projects for approximately 15 years. He supports big companies in defining and organizing the contractual issues of software projects. On the other side, we spoke with a supplier with experience in software projects for approximately 20 years. He is an owner of a software development company with 10 programmers. Considering the sensitivity of failure research and the resulting difficulty in gaining access to project details, this methodology was most appropriate. The incomplete script of the semi-structured interview format left room for improvising questions (Myers and Newman, 2007). The first interview lasted approximately 3 hours; the second lasted 1.5 hours. We made extensive notes during the interviews, which we evaluated afterward through a qualitative content analysis. Because we demanded appointed circumstances and facts, we avoided free interpretation problems (Gläser and Laudel, 2009).

3.1 Results from the Online Survey

The survey showed that the proportion of fixed-price contracts for software development projects is extremely high (Figure 5). Taking into account that even the so-called *agile fixed-price*, *time-and-material (T&M) price with ceiling* ultimately determines the maximum total budget for the consumer, the proportion of this type of contract is a total of more than three quarters of the software development projects. A manager on the side of the supplier added in free text: “Even if it is charged at T&M, the expectation of the customer is the compliance with the budget / value of the order.”

On the bottom line, the T&M price with ceiling and the agile fixed-price mean the implementation of the requirements at fixed cost. Often the ceiling does not differ significantly from the calculated expense. An agile fixed-price, however, allows one to the implementation of requirements when new requirements emerge. Then, these new requirements can replace earlier ones. However, such contractual subtleties relate only to new requirements. A third party (judge) can evaluate them. Nevertheless, this rarely helps in cases of closing the requirement gaps. Rather, closing gaps only makes unconscious knowledge aware. For the customer, it appeared typically obvious, whereas it was unknown to the supplier and vice versa. Filling the gaps makes it known explicitly.

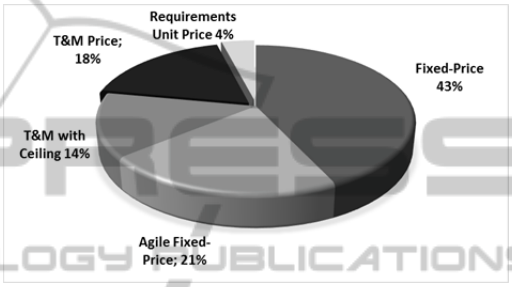

Figure 5: Proportion of different types of contracts on software development projects.

The customers predominantly determine the contract model (Figure 6). Although 80% of the customers indicate that they at least often determine the contract model, suppliers say quite the opposite. Two-thirds of them admit that they have little or no influence on the contract model. One comment from a project leader on the supplier side is: “I do not understand the question. The contract model is in all cases defined by the customer.” Thus, customers clearly choose the contract design.

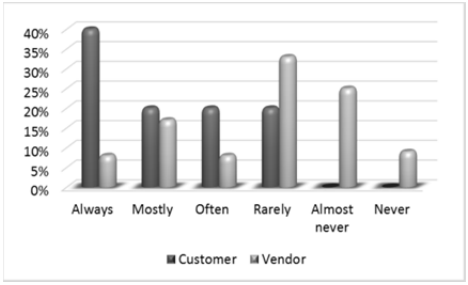

Figure 6: Answer to the question "Do you determine the contract model?".

Customers and suppliers have different views on emerging problems inside a fixed-price project, like when an imbalance occurs in terms of time, cost, and quality (Figure 7).

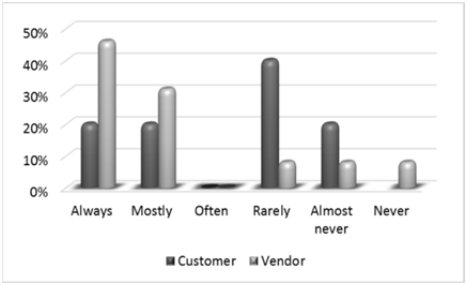

Figure 7: Is an imbalance of time, cost, or quality in the project under fixed-price problematic?.

Although 77% of the suppliers consider such a situation always or usually as problematic, 60% of the customers believe that this is rarely or almost never a problem for them.

Against this background, it is important to consider how the contract reflects gaps in the requirement specifications and how the signed contract supports the project itself. After all, such gaps lead to increased interaction. Most respondents stated for the vast number of projects (Figure 8) that such gaps exist.

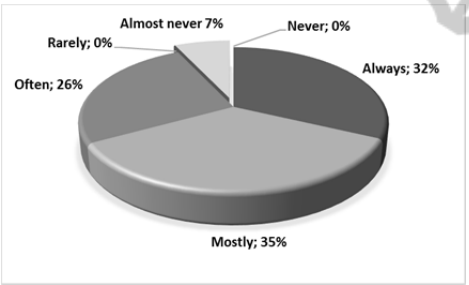

Figure 8: Frequency of requirement gaps.

Almost a third of the respondents said that such gaps “always” happen; 93% say that this case occurs at least often. However, a fixed-price contract hardly takes this sufficiently into account. For suppliers to do this seems hardly to be possible, as the notes to the relevant questions show. They try to work with a kind of overhead calculation but requirement gaps "are rarely sufficiently taken into account."

However, contracts widely do not reflect this fact. On the question, whether contractors continuously update the contract during the project, 81% of participants responded that this rarely or never happens.

Customers and suppliers have a different perspective regarding whether gaps leading to unforeseen interaction would be renegotiated (Figure 9). Although customers are of the opinion that this would always or at least often happen, 61% of the suppliers believe that there are never or almost never renegotiations.

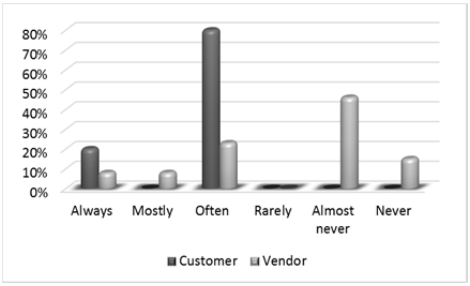

Figure 9: Renegotiate customer and supplier requirement gaps.

Two-thirds of all respondents say that gaps in the requirement specifications always or almost always lead to unplanned discussions. The contract usually does not take into account the extra costs, which interactions trigger.

3.2 Results from Interviews

We documented the interviews in a structured way with references to each question and to the paragraph of the answer. In the following, we give a short overview of the results. In brackets, we note the reference to the minutes of the interviews. For example, (S Q3A2) references the supplier interview, question 3, answer paragraph 2.

Both interview partners said that the mostly preferred contract model is the fixed-price contract, especially if the requirements are documented and if they seem to be clear (C Q3A1, S Q5A2). This is because of the customer’s restriction in having a limited budget and that customers must calculate the expected benefits against the costs beforehand (C Q16A1, (S Q6A1). Nevertheless, because “it is very seldom that the requirements are specified in a formal way” (C Q10A1), it is almost impossible to calculate the real costs. In addition, the supplier stated: “The problem does not come from the fixed-price itself, but from unclear, incomplete, or changing requirements. And the problem is that the customer is not willing to change the price if he changes the requirements” (S Q6A3).

The interviews supported the finding from the online survey, that the customers mostly dominate the contract design (S Q5A2, C Q3A1). Nevertheless, both interview partners gave hints, that obligations for a cooperating behavior of the customer are possible in practice (C Q14A4, S Q11A6).

Because the requirement specifications were so important, we asked our interview partners to explain the reasons for the gaps, the possibilities for dealing with these gaps, as well as the consequences. Both sides cited the reasons as being "special" or "exceptional use cases" that the experts were not aware of during the requirements analysis or were too difficult to model (C Q11A1; C Q11A4; S Q10A1). Furthermore, the facts were "obvious" (C Q11A3) or "self-evident" (S Q10A1) to the business experts, so they did not speak about them. Nonfunctional requirements were often unknown to the users (S Q10A1).

Both interview partners showed a high degree of uncertainty regarding the behavior, intentions, and skills of the other side. Customers try to get certainty beforehand from information like "descriptions of credential projects, facts about the know-how of their staff, information about the methods in designing and processing a software project" (C Q7A1). With "governance structures for the project" (C Q5A1) the customer hopes to "get at early phases of the project a good feeling of the progress and the quality of the vendor's work" (C Q6A1). However, uncertainty remains high: "Nearly nobody can distinguish the clever, good one from the slow and poor one. And if the vendor mentions that there are unforeseeable problems, you don't know if he is right or he is not professional enough for doing his job" (C Q16A1).

Regarding the same issue, the interview partner from the supplier side said, "a new management, problems in his market, new relevant law, and maybe, the customer does not need the software anymore or the costs will be higher than the effects. Then, maybe, the customer's management tries to cancel the project" (S Q11A5).

On the customer side, the strategy is to handle all problems in a formal way and to avoid all discussions regarding effort in narrowing the gaps in the requirements (C Q11A5; C Q15A1). In contrast, the supplier obviously has strategies of its own, knowing that the customer cannot see all that the supplier is doing (S Q12A1).

4 CONCLUSIONS

The objective of this paper was to describe the software development project as an interaction between two organizations, both acting as rational agents, both having economic targets. We have shown that these actors are in a dilemma situation, known from game theory as the prisoner's dilemma. In such a situation, the individual rational behavior of both actors leads to a result that does not satisfy either parties—neither the customer nor the supplier.

The root cause of this situation is the incompleteness of the requirement specifications. As theoretical and empirical investigations show, a specification without gaps is not possible. Therefore, the parties must cooperate when closing the gaps. Nonetheless, particularly under the most widely used fixed-price contract, both parties must avoid efforts in this cooperation.

Certainly, our investigation is not representative. However, our aim was to support our theoretical findings. As our survey shows, the customer often dominates the contractual regulations. In this situation, the customer can avoid the effort in closing the requirement gaps, whereas the supplier is forced to cooperate. As a rational agent, the supplier will use information asymmetries to save effort by hidden actions. This results in a poorly developed software system. Based on our two-party model, future research can analyze the dependencies of asymmetrically distributed information and software quality.

Using the game theory, we can describe the problem, but we can also show the way out. We can derive from the model the suggestion to connect defection with a sanction, and therefore change the situation. Defining the obligations for closing the requirement gaps for both the customer and the supplier within the contract can serve as such a sanction. We suggest that customers and suppliers agree on clear and tangible obligations for the customer regarding the cooperation for filling the gaps in requirement specifications. These contractual obligations should contain information on the necessary staff and the time required. Then, if the customer fails to meet these obligations, the parties may agree on a bonus for the supplier to be offset with possible penalties. In further research, we can include the theory of incomplete contracts.

Furthermore, we can use the results from research about the prisoner's dilemma (Axelrod, 2009). If both parties are willing and able to cooperate, then it can be rational to start interactions with cooperation. In this way, both sides need a system to recognize and measure the behavior of the other party. Because experience is a prerequisite for trust, further research should examine whether the methods and concepts in the software development project are suitable for the formation of experience. We can derive such concepts from approaches of economic theories using the theoretical descriptions of customer and supplier as rational agents.

REFERENCES

Al-Ahmad, W., Al-Fagih, K., Khanfar, K., Alsamara, K., Abuleil, S., Abu-Salem, H., 2009. A Taxonomy of an IT Project Failure: Root Causes. *International Management Review* 5(1), 93-104.

Axelrod, R., 2009. *Die Evolution der Kooperation*, Studienausgabe, München. 7nd edition.

Beckmann, M., Pies, I., 2006. Freiheit durch Bindung - Zur ökonomischen Logik von Verhaltenskodizes, *Diskussionspapier Nr. 2006-9*, Lehrstuhl für Wirtschaftsethik der Martin-Luther-Universität Halle.

Buhl, H.U., Meier, M.C., 2011. Die Verantwortung der Wirtschaftsinformatik bei IT-Großprojekten. *Wirtschaftsinformatik* 2, 59-62.

Chua, C.E.H., Lim, W.-K., Soh, C., Sia, S.K., 2012. Client strategies in vendor transition: A threat balancing perspective. *The Journal of Strategic Information Systems* 21(1), 72-83.

Cockburn, A., 2004. The End of Software Engineering and the Start of Economic-Cooperative Gaming. *ComSIS* 1(1).

El Emam, K., Koru, A.G., 2008. A Replicated Survey of IT Software Project Failures. *IEEE Software* 25(5), 84-90.

Davis, L.H., 1985. Prisoners, Paradox, and Rationality. Paradoxes of Rationality and Cooperation. in: Campell, R. and Sowden, L. (eds), *Prisoner's Dilemma and Newcomb's Problem*. Vancouver, 46-59, Reprint of *American Philosophical Quarterly* 14, 4, 1977, 319-327.

Dwivedi, Y.K., Ravichandran, K., Williams, M.D., Miller, S., Lal,B., Antony, V., Muktha, K., 2013. IS/IT Project Failures: A Review of the Extant Literature for Deriving a Taxonomy of Failure Factors. *IFIP Advances in Information and Communication Technology* 402, 73-88.

Fink, L., Lichtenstein, Y., Wyss, S., 2013. Ex post adaptions and hybrid contracts in software development services. *Applied Economics*. 45(32), 4533-4544.

Gauthier, D., 1985. Maximization Constrained: The Rationality of Cooperation. Paradoxes of Rationality and Cooperation. in: Campell, R. and Sowden, L. (eds) *Prisoner's Dilemma and Newcomb's Problem*, Vancouver, 75-93.

Gläser, J., Laudel, G., 2010. *Experteninterviews und qualitative Inhaltsanalyse.* VS Verlag, Wiesbaden, , 4nd edition

Hazzan, O., Dubinsky, Y. 2005. Social Perspective of Software Development Methods: The Case of the Prisoner Dilemma and Extreme Programming. *Lecture Notes in Computer Science 3556*, 74-81.

Kalnins, A., Mayer, K.J., 2004. Relationships and hybrid Contracts: An Analysis of Contract Choice in Information Technology. *Journal of Law, Economics, and Organization*, 20(1), 207-229.

Kano, N., Seraku, N., Takahashi, F., Tsuji, S., 1984. Attractive Quality and Must Be Quality; in: *Quality - Journal of the Japanese Society for Quality Control*, 14(2), 39-44.

Keil, M., Smith, H.J., Pawlowski, S., Jin, L., 2004. 'Why Didn't Somebody Tell Me?': Climate, Information Asymmetry, and Bad News About Troubled Projects. *SIGMIS Database*, 35(2), 65-84.

Klammer, B., 2005. *Empirische Sozialforschung. Eine Einführung für Kommunikationswissenschaftler und Journalisten*. Utb, Konstanz.

Liu, J.Y.-C., Chen, H.-G., Chen, C.C., Sheu, T.S., 2011. Relationships among interpersonal conflict, requirements uncertainty, and software project performance.

McGee, S., Greer, D., 2012. Towards an understanding of the causes and effects of software requirements change: two case studies. *Requirement Engineering* 17, 133-155.

Mayer, H., 2012. *Interview und schriftliche Befragung. Entwicklung, Durchführung und Auswertung.* Oldenbourg Wissenschaftsverlag, München.

Myers, M. D., Newman, M., 2007. The qualitative interview in IS research: Examining the craft. *Information and Organization* 17(1), 2-26.

Natovich, J., 2003. Vendor Related Risks in IT Development: A Chronology of an Outsourced Project Failure. *Technology Analysis & Strategic Management* 15(4), 409-419.

Rustagi, S., King, W.R., Kirsch, L.J., 2008. Predictors of Formal Control Usage in IT Outsourcing Partnerships. *Information Systems Research* 19(2), 126-143.

Schnell, R., Hill, P., Esser, E., 2011. *Methoden der Sozialforschung.* Oldenbourg Wissenschaftsverlag, München, 9nd edition.

Spremann, K., 1990. Asymmetrische Information. *ZfB* 60(5/6), 561-586.

Standish Group, 1995. CHAOS Report, http://www.projectsmart.co.uk/docs/chaos-report.pdf, 21.06.2011.

Standish Group, 2010. CHAOS MANIFESTO, The Laws of Chaos and the CHAOS 100 Best PM Practices. https://secure.standishgroup.com/reports/reports.php#reports. accessed 26.06.2011.

Tollefsen, D., 2002. Organizations as true believers. *Journal of social philosophy* 33(3), 395-410.

Tucker, A.W. 1950. Biographie, Prisoner's Dilemma. http://www.princeton.edu/pr/news/95/q1/0126tucker.html.

Yilmaz, M., O'Connor, R.V., Collins, J., 2010. Improving Software Development Process through Economic Mechanism Design Communications. *Computer and Information Science* 99, 177-188.

Zannier, C., Maurer, F., 2007. Comparing Decision Making in Agile and Non-agile Software Organizations. Concas, G. et al. (eds). *XP 2007, LNCS* 4536, 1–8.