The Impact of Software Development Companies on Software Engineers' Responses to Incomplete Requirements

Ozlem Albayrak, Duygu Albayrak
Department of Computer Technology & Information Systems
Bilkent University
06800 Bilkent, Ankara Turkey
ozlemal@bilkent.edu.tr

من الشائع القبول بأن جودة المتطلبات للبرنامج تؤثر على جودة البرنامج المنتج. والبرامج ذات الجودة العالية تعتمد على متطلبات كاملة. أما بالنسبة للمتطلبات الناقصة فتعتمد على مايحاول مهندس البرامج من تعبئته من متطلبات غير مكتملة وذلك إما بأخذها من ردود المستخدمين أو بالافتراض. هذه الافتراضات قد تكون صريحة أو ضمنية. الافتراضات الصريحة عادة ماتكون أفضل من الضمنية لأن الفرضيات الصريحة بالاستطاعة التأكد من صحتها. لقد قمنا بإجراء دراسة تجريبية لمعرفة إذا ماكان عدد الافتراضات الصريحة من قبل المهندسين مرتبطة بعدد الشركات التي كانوا يعملون بها. لقد قمنا بالتحري عن طريق استخدام بيانات من ثمان شركات تمثل ردود 251 من مهندسي البرامج لنفس متطلبات البرامج الناقصة. نتائج هذه الدراسة تبين علاقة قوية بين شركة تطوير البرامج وعدد الافتراضات الصريحة من قبل المهندسين الذين يعملون لتلك الشركة.

The Impact of Software Development Companies on Software Engineers' Responses to Incomplete Requirements

Ozlem Albayrak, Duygu Albayrak
Department of Computer Technology & Information Systems
Bilkent University
06800 Bilkent, Ankara. Turkey
ozlemal@bilkent.edu.tr

***ABSTRACT*:** *It is commonly accepted that software requirements quality affects software product quality, and high-quality software products depend on complete requirements. With incomplete requirements, depending on the requirement software engineers attempt to fill gaps differently; either by getting feedback from the user or by making assumptions. Assumptions may be explicit or implicit. Explicit assumptions are preferable to implicit assumptions because explicit assumptions can be validated. We conduct an empirical study to determine whether the number of explicit assumptions made by software engineers is related to the companies that the engineers work for. Using data from eight companies we investigate the responses of 251 software engineers to the same incomplete software requirement. The results of the study show a significant relationship between a software development company and the number of explicit assumptions made by the engineers who work for that company.*

Keywords: Software developments, Software quality

Received: 1 June 2009, Revised 12 July 2009, Accepted 18 July 2009

1. Introduction

The quality of a software product is an important factor in the success of a software project. Although every software organization aims to develop software that meets functional needs with acceptable levels of quality within budget, and on schedule (Agrawal, 2007; Noppen, Broek & Aksit 2008), only some of them succeed. Deficient and low-quality requirements may be the major reason for software project failures (Cheng & Atlee, 2007; Zagjsek & Separovic, 2007; Hoffman & Lehner, 2001; Saiedian & Dale, 2000; Kamata & Tamai, 2007; Rowen, 1990; Yeh & Zave, 1980). Quality of software requirements is measured by eight attributes: Correctness, unambiguity, completeness, consistency, ranking, verifiability, modifiability, and traceability (IEEE, 1998, Agrawal, 2007). Complete and correct requirements specifications are required for developers to know what to build and for users to know what to expect (Saiedian & Dale, 2000; Rowen, 1990).

In this study, we focus on the completeness attribute of software requirements. A complete software requirement should contain all necessary information, including constraints and conditions. When software engineers face incomplete requirements, they attempt to fill the gaps by information from the stakeholders or by assumptions. When information from the stakeholders is not available, the engineers make assumptions. Their assumptions may be explicitly stated, and thus documented, or implicitly carried further, to the design and implementation phases, where there is a high probability that the related requirements will be incorrect. For high-quality requirements, software engineers should aim to fill software requirements' gaps with explicit assumptions. The problems caused by implicit assumptions may be alleviated, to some degree, if the assumptions are made explicitly.

This study aims to determine possible relationships between software engineers' tendencies to make explicit assumptions and the software development companies that the engineers work for. If reasons for making implicit assumptions are found to be company related, ways to avoid such assumptions may be better identified.

The following sections address background information on requirements engineering (RE) and assumptions made to fill in incomplete requirements, as well as detailed information on the study (the research question, sample, and method), empirical findings and data analysis from a series of projects, threats to validity, and conclusion and future directions.

2. Background: RE and assumptions

This section presents background information on RE and assumptions made by software engineers in the case of incomplete requirements.

2.1 Requirements Engineering

An effective requirements process at the beginning of a project has positive outcomes throughout the project's life cycle, improving the efficacy of other project processes and ultimately leading to improvements in many aspects, including in product quality (Damian & Chisan, 2006).

Software requirements engineering is defined as all the activities denoted to identify user requirements to drive additional requirements, document the requirements as a specification, and validate the documented requirements against the actual user needs (Saiedian & Dale, 2000). The goal of RE is to assure that an effective and high-quality product is defined and developed from the stakeholders' point of view (Dörr, Adam, Eisenbarth & Ehresman, 2008; Kujala, Kauppinen, Lehtona & Kojo, 2005). RE is a time-consuming and expensive, but critical, phase in software (and system) development (Mannion & Keepence, 1995).

Requirements elicitation is composed of activities that enable understanding the goals, objectives, and motives for building a proposed system (Cheng & Atlee, 2007). Ways to perform successful RE activities have been studied (Davis, Dieste, Hickey, Juristo & Moreno, 2006; Boehm, 1996; Saeidian & Dale, 2000; Willoughby, 1989) and many different techniques and approaches related to elicitation have been determined (Hoffman and Lehner, 2001; Rowen, 1990; Llyod, Rosson & Arthur, 1995; Subramaian, 1999).

Regardless of the type of elicitation techniques, user involvement is an important element. Kujala et al. studied the role of user involvement in RE quality and project success and concluded that early user involvement seems to be a powerful way of improving requirements quality and project success (Kujala, Kauppinen, Lehtona and Kojo, 2005). Better-quality requirements can be developed when they are generated by ongoing client interaction, with a constantly improving prototype to reduce ambiguity (Schrage, 2004; Albayrak, Albayrak & Kilic, 2009; Redondo, Arias, Martinez & Vilas, 2002). Users must be carefully listened to and implicit assumptions must never be made (Saiedian & Dale, 2000), as such assumptions are not shared with stakeholders and thus may increase the uncertainty of the requirements (Ebert and Man, 2005; Strens & Sugden, 1996).

Despite its importance, the "imperfect or incomplete information" problem has not been satisfactorily studied in the software engineering literature (Noppen, Broek and Aksit, 2008). Insufficient attention paid to RE results in myriad problems, such as rework, validity problems, regarding incomplete requirements (Subramanian, 1999). RE process-improvement methods typically work with explicit process models with explicit document definitions (Doerr, Paech & Koehler, 2004). If incomplete requirements are unavoidable, we should definitely avoid accepting them as complete by using implicit assumptions.

3. Assumptions Made by Software Engineers

We name the missing information between complete and incomplete software requirements as the "requirement gap." When engineers make assumptions explicitly, they are aware of which gap they fill and how they fill it. Explicit assumptions enable engineers to share their assumptions with users.

In the case of implicit assumptions, most software engineers do not even realize that they are making assumptions. They perceive the requirement as complete and continue software development with their perceived requirements rather than with the users' complete requirements. When software engineers fill the gaps with information not recorded and shared, and hence not confirmed by the user, they create virtual requirements rather than actually filling the gap. These virtual requirements often result in false requirements, which may be the primary source of user change requests, rework, validity problems, and even project failure.

Recent studies have shown some evidence that the more implicit assumptions are made by engineers the more volatile the resulting requirements (Rowen, 1990). Albayrak et al. (Albayrak, Albayrak & Kilic, 2009) studied project-related factors, one of many variables that may impact whether software engineers make explicit or implicit assumptions. Within the limitations of the study, the results supported the propositions that the type of client, current phase of software development life cycle (whether it includes an analysis phase or not), tools utilized for RE, and RE training taken during the project development influence whether the engineers make explicit or implicit assumptions. According to the study results, software engineers make more explicit, hence less implicit, assumptions when:

- the client is military,
- the analysis is part of the current phase of the software development life cycle,
- the tools for RE are utilized, and
- the engineers have recently undergone RE training.

In addition to the aforementioned supported propositions, the study investigated whether the propositions related to project size and existence of subcontractors was supported (Albayrak, Albayrak & Kilic, 2009). The propositions claiming that

relatively larger projects have greater explicit assumptions and that the existence of subcontractors may increase the tendency to make more explicit assumptions were not supported.

Another study aimed to determine if the ways software engineers preferred to respond to an incomplete requirement (by prototype, pseudo code, source code), their working experience and/or their educational background impacted the type of assumptions made (Albayrak, Kurtoglu & Bicakci, 2009). According to the results of that study, there is a significant relationship between the engineers' experience and the number of explicit assumptions made. The study results also revealed that on average, non-computer-background engineers made more explicit assumptions than computer-background graduates.

Identifying the factors that determine software engineers' preferences for filling information gaps is a challenging subject to study. Studies have been recently conducted on project-related and software-engineer-related factors (Albayrak, Bicakci & Bozkurt, 2009; Albayrak, Albayrak & Kilic, 2009; Albayrak, Kurtoglu & Bicakci, 2009); this study observes the possible impact of company-wide factors on the type of assumptions that software engineers make in the case of incomplete requirements.

4. The Study

This section provides the pertinent information about our experiments: The research question, the sample, and the method used in the study.

4.1 The Research Question

The primary research question of this study is to investigate whether software engineers' responses to incomplete requirements are related to company factors.

The major research question of the study was:

Is the number of explicit assumptions made by software engineers impacted by the software development company that they currently work for?

To investigate the above question, the following hypotheses were defined. The null hypothesis, $H1_0$, is followed by the alternative hypothesis, H1a.

$H1_0$: The number of explicit assumptions made by software engineers is impacted by the software development company that they currently work for.

H1a: The number of explicit assumptions made by software engineers is not impacted by the software development company that they currently work for.

We define an average number of explicit assumptions made by software engineers per company as the study's dependent variable.

The independent variable is the software development company that an engineer worked for at the time the study was conducted. None of the engineers involved in the study worked in more than one company at a time. Thus, for each participant we had only one value for the company data.

We assume that the company may have an impact on the number of explicit assumptions made by engineers. We expect to observe such differences due to company-wide variables regarding the tendency of engineers to make explicit assumptions.

4.2 Sample and Method

In this empirical study, we conducted an experiment and collected data regarding software engineers' preferences in completing a given deficient software requirement [Appendix A] of Albayrak, Albayrak & Kilic, 2009. For each company, we calculated the mean of explicit assumptions made by the engineers. During the experiment, software engineers' access to stakeholders was restricted.

In the study, we first conducted pre-interviews with software development directors of the companies. The directors later submitted the question used in (Albayrak, 2009), and selected project managers who directed the question to the software engineers. The collected data was sent to us from the directors. In examining the data, we first counted explicitly written questions and assumptions as explicit assumptions. We identified eight common gap types related to the given requirement. Gap types other than those listed in (Albayrak, 2009) are found, but with very low frequency. Records including different gap types were also counted in the experiment data. For each company, we calculated the mean of explicit assumptions made. After compiling data collected from the software engineers, we conducted post-interviews.

During the post-interviews, the possible elements regarding company factors were discussed both with the managers and with the engineers who participated in the study. All company data was collected by the same author during the experiment.

5. Results and Analysis

Prior to conducting an ANOVA, we first calculated descriptive statistics. Using descriptive statistics data and presenting box plot analysis, we first compared descriptive statistics regarding the sample data. Table 1 shows mean%, mean, sample size N, and standard deviation of the average number of explicit assumptions made by software engineers working at the companies involved in the study.

Engineers of companies C5 and C8 made the highest mean number of explicit assumptions, while those working at C4 and C3 collected the minimum mean number of explicit assumptions (Table 1). Post-interviews were conducted at these four companies.

Ci: Company	Mean %	Mean	N	Std. Deviation
C1	19	1,52	50	,8862
C2	19,75	1,58	19	1,1698
C3	5,50	,44	45	,7247
C4	2,75	,22	9	,4410
C5	35,63	2,85	40	2,1668
C6	7,25	,58	19	,7685
C7	11,63	,93	27	1,4657
C8	29,13	2,33	42	1,8033
Total	18,75	1,50	251	1,6282

Table 1. Descriptive Statistics

The box plot in Fig.1 shows the results graphically. The thick line represents the mean value, the whiskers indicate the extreme points excluding the outliers, and the box contains the middle 50% of the data.

We used one independent variable (company) and one dependent variable (number of explicit assumptions) in the study. We conducted a one-way analysis of variance ANOVA to evaluate the effect of company data on the number of explicit assumptions made (Table 2).

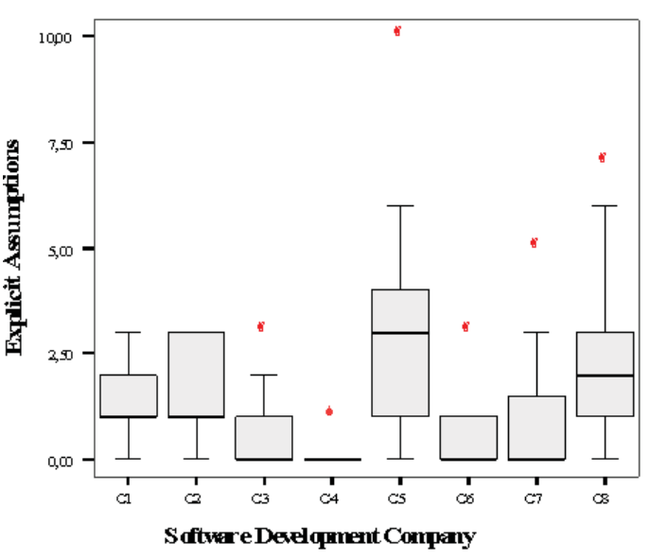

Figure 1. Company versus explicit assumptions

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	192.054	7	27,436	14.164	0.000
Within Groups	470.695	243	1.937		
Total	1226.0	51			

R Squared = 0.290 (Adjusted R Squared=0.269)

Table 2. Test of the ANOVA

The ANOVA was significant: F(7,243) = 14,164 and p = 0.000. 29.0% of variance in the number of explicit assumptions made is explained by the company factor. Findings of our study revealed that there is a significant relationship between the company and the average number of explicit assumptions made by the company's software engineers.

The results of the post-interviews suggested that the impact caused by the company variable should be studied using different attributes of company. Among the suggested attributes, we plan to collect data related to software development processes, e.g. CMMI level of the companies, standards used by the company regarding software development, size of the company and whether the company is an international establishment.

6. Threats to Validity

As with any empirical study, there are various threats to validity that must be discussed. In this section we discuss the internal and external validities of our study. Internal validity is defined as the soundness of the conceptual relationships within a study.

The first threat is the threat of subject characteristics (or selection bias). We selected a convenience sample. We had no control over the selection of the subjects. The specific subjects who participated in the study could be the major reason for the observed results

The second threat to the internal validity of this study is the threat of data-collector characteristics. At each company, different collectors collected data from the subjects. The characteristics of the data collectors might have affected the results. In addition, the data collector may have unconsciously distorted the data in such a way as to make certain outcomes more likely, leading to a data-collector bias threat.

External validity is defined as the degree to which results from the study can be generalized and provide insight. The representativeness of the artifact is a threat to external validity. We used a very simple, textbook-sample-like artifact previously used in (Albayrak, 2009; Albayrak, Bicakci & Bozkurt, 2009; Albayrak, Albayrak & Kilic, 2009; Albayrak, Kurtoglu & Bicakci, 2009). We selected this generic (not domain-specific) artifact because we wanted to make sure that all the subjects were equally familiar with the requirement. Since it was simple, it did not take much time for the subjects to complete. The artifact used in this study may not be reflective of an actual requirements document. We consider using a more realistic instrument for future studies.

The last threat is common to all empirical studies. It cannot be assumed that the results will always generalize beyond the setting in which the study was conducted. Thus, for more confidence in the results, the study should be replicated. After the post-interview data analysis, we started a study to collect other attributes related to the company variable to observe if they impact the dependent variable.

7. Conclusion and Future Studies

How gaps in software requirements are filled by software engineers is important, and depending on the method used, may lead to project failure. This study focuses on the possible and not previously studied relationships between company-related factor and use of explicit assumptions by software engineers. We construct a base for future studies looking for possible relationships between various factors and the number of explicit assumptions made by the software engineers in the case of incomplete requirements. We observed a significant impact of company factor on the dependent variable.

Factors impacting software engineers' preferences to fill gaps may not be limited to company and project-related specifications. Engineer-related factors, and interrelations to project- and company- related factors may also be studied in future. For further studies, parameters of organizational and software-engineer-related characteristics may be included.

In addition to functional attributes, quality attributes are also very crucial to the success of software projects (Cheng & Atlee, 2007). Further studies may also focus on incomplete quality-attributes-related requirements.

When the relationships between software engineers' preferences in completing deficient requirements and project-related parameters are identified, actions should be taken to prevent implicit assumptions. Building prototypes may help companies to minimize the number of implicit assumptions made.

References

[1] Agrawal, M. and Chari, K. (2007). Software Effort, Quality, and Cycle Time:A Study of CMM Level 5 Projects, *IEEE Transactions on Software Engineering*, *33*(3), 145-156.

[2] Albayrak, O. (2009). Solutions to Challenges of Teaching Systems Analysis and Design for Undergraduate Software Engineers, In *System Analysis and Design for Advanced Modeling Methods: Best Practices,* Akhilej Bajaj and Stanislaw Wrycza (Eds), 68-87, IGI Global.

[3] Albayrak, O., Bicakci, M. and Bozkurt, H. (2009). A Study to Observe Relations Between Software Engineer's Responses to Incomplete Requirements and Requirements Volatility, *International Conference on Software Engineering Theory and Practice*, SETP 2009, Orlando, 1-7.

[4] Albayrak, O., Albayrak, D. and Kilic, T. (2009). Are Software Engineers' Responses to Incomplete Requirements Related to Project Characteristics?, *Proceedings of the Second International Conference on the Applications of Digital Information and Web Technologies* (ICADIWT 2009), London, 188-195.

[5] Albayrak, O., Kurtoglu, H., and Bicakci, M. (2009). Incomplete Software Requirements and Assumptions Made by Software Engineers, *The 16th Asia-Pacific Software Engineering Conference*, (APSEC 2009), Penang, (accepted paper).

[6] Boehm, B. (1996). Identifying Quality-requirement Conflicts, *IEEE Software*, *13*(2), 25-35.

[7] Cheng, B.H.C. and Atlee, J.M. (2007). Research Directions in Requirements Engineering, *Future of Software Engineering* (FOSE '07), IEEE 2007, 285-303.

[8] Damian, D. and Chisan, J. (2006). An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management, *IEEE Transactions on Software Engineering*, *32*(7) 433-453.

[9] Davis, A. Dieste, O, Hickey, A., Juristo, N. and Moreno, A.M. (2006). Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived From a Systematic Review, *Proceedings of the IEEE Int. Req. Eng. Conf.* (RE), 176-185.

[10] Doerr, J., Paech B., and Koehler, M. (2004). Requirements Engineering Process Improvement Based on an Information Model, *Proceedings of International Conference on Requirements Engineering* (RE04), IEEE Computer Society Press, Los Alamitos, USA, 70-79.

[11] Dörr, J., Adam, S., Eisenbarth, M., and Ehresman, M. (2008). Implementing Requirements Engineering Processes: Using Cooperative Self-Assessment and Improvement, *IEEE Software*, 25(3), 71-77.

[12] Ebert, C., and Man, D. (2005). Requirements Uncertainty: Influencing Factors and Concrete Improvements, ICSE'05, *Proceedings of International Conference on Software Engineering*, 553-560.

[13] Hoffman, H.F. and Lehner, F. (2001). Requirements Engineering as a Success Factor in Software Projects, *IEEE Software*, 58-66.

[14] IEEE Std. 830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE.

[15] Kamata, M.I., and Tamai, T. (2007). How Does Requirements Quality Relate to Project Success or Failure?, *Proceedings of the 15th International Requirements Engineering Conference*, IEEE, 69-78.

[16] Kujala, S., Kauppinen, M., Lehtona, L., and Kojo, T. (2005). The Role of User Involvement in Requirements Quality and Project Success, *Proceedings of the 13th IEEE International Conference on Requirements Engineering* RE(2005), 75-84.

[17] Lloyd, W.J., Rosson, M.B., and Arthur, J.D. (2002). Effectiveness of Elicitation Techniques in Distributed Requirements Engineering, *Proceedings of IEEE Joint International Requirements Engineering Conference on Requirements Engineering*, 311-318.

[18] Mannion, M., and Keepence, B. (2005). Smart Requirements, *ACM Software Engineering Notes*, 20, p.42.

[19] Noppen, J., Broek, P. and Aksit, M. (2008). Software development with imperfect information, *Soft Comp*, 3-28.

[20] Redondo, R.P.D., Arias, J.J.P., Martinez, A.F., and Vilas, B.B. (2002). Approximate Retrieval of Incomplete and Formal Specifications Applied to Vertical Reuse, *Proceedings of International Conference Software Maintenance*, 618-627.

[21] Rowen, R.B. (1990). Software project management under incomplete and ambiguous specifications, *IEEE Transactions on Engineering Management*, *37*(1), 10–21.
[22] Saiedian, H. and Dale, R. (2000). Requirements Engineering: Making the Connection Between the Software Developer and Customer, *Information and Software Technology*, 42, 419-428.
[23] Schrage, M. (2004). Never go to a client meeting without a prototype [software prototyping], *IEEE Software*, *21*(2), 42-45.
[24] Strens, M.R., and Sugden, R.C. (1996). Change Analysis: A Step towards Meeting the Challenge of Changing Requirements, *Proceedings of the IEEE Symposium and Workshop on Engineering of Computer Based Systems*, p.278.
[25] Subramanian, U.V. (1999). An Event, Activity and Process Based Methodology for Requirements Elicitation and Its Application to an Educational Information System, *Proceedings of the Sixth Asia Pacific Software Engineering Conference*, (APSEC '99), 188-195.
[26] Willoughby, J.K. (1989). Adaptations to the Systems Engineering Management Process for Projects with Incomplete Requirements, *Proceedings of IEEE International Conference on Systems Engineering*, 197-200.
[27] Yeh, R.T., and Zave, P. (1980). Specifying Software Requirements, *Proceedings of the IEEE, 68*(9), 1077- 1085.
[28] Zagajsek, B. Separovic, K., and Car, Z. (2007). Requirements Management Process Model for Software Development Based on Legacy System Functionalities, *9th International Conference on Telecommunications*, (ConTel 2007), 115-122.