Are Software Engineers' Responses to Incomplete Requirements Related to Project Characteristics?

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

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

Tuna Kılıç
STM, Ankara Technology Development Region
5th Avenue No:2/3,
06800 Bilkent, Ankara/ Turkey
tkilic@stm.com.tr

Abstract

Software requirements quality affects software product quality. For high-quality software products, software requirements must be complete. When faced with incomplete requirements, software engineers attempt to fill the requirements' 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 as 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 a project's characteristics. Using data from two CMMI Level 3 companies and 16 projects, we investigated the responses of 92 software engineers to the same incomplete software requirement. Our findings show possible relationships between projects' characteristics and the number of explicit assumptions.

1. Introduction

Every software organization aims to develop software that meets functional needs with acceptable levels of quality, within budget, and on schedule [13]. Unfortunately, not all succeed. Deficient and low-quality requirements may be the major reason for software project failures [3, 4, 7, 8, 14, 21, 23]. Software requirements quality is related to correctness, unambiguity, completeness, consistency, ranking, verifiability, modifiability, and traceability [9, 13].

In this study, we concentrate on the completeness attribute of software requirements. The 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. The assumptions may be explicitly stated or implicitly carried further, to the design and implementation phases.

Software engineers should aim to minimize filling software requirements' gaps with implicit assumptions because there is a high probability that the assumptions will be incorrect. Complete and correct requirements specifications are required for developers to know what to build and for users to know what to expect [8, 21].

This study aims to determine possible relationships between software engineers' tendencies to make explicit assumptions and project characteristics. If reasons for making implicit assumptions are found, ways to avoid them may be better determined.

The following sections address background information on requirements engineering; 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. Requirements engineering

An effective requirements process at the beginning of the project has positive outcomes throughout the project life cycle, improving the efficacy of other project processes and ultimately leading to improvements in many aspects, including product quality [6].

Software requirements engineering (RE) 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 [8]. The goal of RE is to assure that an effective and high-quality product is defined and developed from the stakeholders' point of view [11, 24].

Requirements elicitation is composed of activities that enable understanding the goals, objectives, and motives for building a proposed system [3]. Ways to perform successful RE activities were studied [1, 2, 8, 12] and many different techniques and approaches related to elicitation were determined [7, 21, 17, 25].

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 [24]. Better-quality requirements can be developed when they are generated by ongoing client interaction, with a constantly improving prototype to reduce ambiguity [16, 22, 22]. Users must be carefully listened to and implicit assumptions must never be made [8], as they are not shared by stakeholders and thus may increase the uncertainty of the requirements [5, 15].

Insufficient attention paid to RE results in myriad problems regarding incomplete requirements [25]. If incomplete requirements are unavoidable, we should definitely avoid accepting them as complete by using implicit assumptions. RE process-improvement methods typically work with explicit process models with explicit document definitions [10].

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.

Identifying the factors that determine software engineers' preferences for filling information gaps is a challenging subject to study.

3. The study

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

3.1. The research question

In our literature survey we did not find a previous study on factors and their relationships to software engineers' preferences for making explicit assumptions. Our study investigates whether software engineers' responses to incomplete requirements are related to project factors.

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

The independent variables are project size, existence of subcontractors, type of client, project's current phase, RE processes tools used, and RE-related training taken during the project's development.

To enhance the statement of the parameters and propositions, we define each parameter and for each parameter we explain our reasoning to the related proposition. p_i denotes the parameters and P_i denotes our propositions.

p_1: (Project size) The software project's size in a planned man-month.

P_1: As the size of the project increases, we expect that software engineers' tendencies to make explicit assumptions will increase; hence the average number of explicit assumptions made regarding incomplete requirements will increase.

The need for formal communication and utilization of standard processes is greater in large projects than in small projects. We believe that the increased need for formal processes may influence software engineers to work more formally and record what they do and why they do it, as well as influence what they assume. Thus, in relatively large projects we expect software engineers to record their assumptions more often and therefore make them explicit.

p_2: (Existence of subcontractor) A binary parameter: If the project involves subcontractors, p_2 is 1, else 0.

P_2: The existence of subcontractors may increase the tendency to make more explicit assumptions.

Subcontractors may create additional points of contact in the projects. Information exchange and recording mechanisms may be more formal when working with subcontractors. We expect that the existence of subcontractors will increase the average number of explicit assumptions made by software engineers.

p_3: (Client type) is a category parameter that can take one value from Military, Civilian (state or private), or International.

P_3: Not all client types require same degree of formality. Clients of the same type may own common attributes. The more formal the client processes are, we expect that the tendency to make explicit assumptions will be greater. We assume that military organizations' levels of formality and standardization are higher than other organizations'. We expect that when the client is a military service, the tendency of engineers to make explicit assumptions will increase.

p_4: (Current phase) A binary parameter that can include any subset of Planning, Analysis, Design, Implementation, Testing, and Maintenance. If the set contains Analysis, p_4 is 1, else 0.

P_4: We expect that engineers working with projects conducting analysis will have a greater tendency to make explicit assumptions. RE processes are part of the analysis phase of a project. People have a tendency to pay attention to their current phase of the project.

p_5: (Tools for RE) A binary parameter that becomes 1 when RE process-related tools are used, 0 otherwise.

P_5: In projects where RE tools are utilized, we expect that software engineers' tendencies to make explicit assumptions will be greater than for projects that do not use RE tools. RE process-related tools help software engineers to better work in formal and improved process environments.

p_6: (RE Training) A binary parameter taking 1 if software engineers undergo RE-related training is during the project, 0 otherwise.

P_6: Our assumption is that if the engineers are involved in RE-related training during their current project, they will care more about requirements quality and have a tendency to make more explicit assumptions.

3.2. Sample and method

This empirical study is composed of two phases. In the first phase, we conducted an experiment and collected data regarding software engineers' preferences in completing a given deficient software requirement (Appendix A). For each project we calculated the mean of explicit assumptions made by engineers. The first phase's sample included six companies and 32 projects. In the second phase, we collect project-related parameters. Some companies did not want to share their project-related data, and the final sample is composed of two companies and 16 projects. Both companies have CMMI Level 3.

In the first phase, we first conducted pre-interviews with software development directors of the companies. The directors later submitted the question used in [18], Appendix A, and selected project managers who directed the question to the software engineers. The collected data was sent to us from the directors. In both companies, interviews were held by the same author.

During the first experiment, we restricted software engineers' access to the stakeholders. We 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 Appendix B are found, but with very low frequency. For each project, we calculated the mean of explicit assumptions made. After compiling data collected from the software engineers, we conducted post-interviews.

In the second phase, a survey, Appendix C, was sent to the project managers via e-mail. The project characteristics were collected by different authors.

3.3. Results and analysis

Using descriptive statistics, we compare means and check whether our propositions were supported. Our findings support four propositions; P_1 and P_2 are not supported.

Table 1. Findings related to p_1: Project Size

p_1: Project size large?	Mean %	Mean	N	Std. Deviation
No	21,68	1,7354	11	0,3606
Yes	19,54	1,5633	3	0,0811
Total	21,23	16985	14	0,2814

The number of explicit assumptions is greater in smaller projects (Table 1).

Table 2. Findings related to p_2: Subcontractor

p_2: Sub-contractor	Mean %	Mean	N	Std. Deviation
No	22,32	1,7858	10	1,2316
Yes	20,69	1,6548	6	0,4151
Total	21,71	1,7366	16	0,9858

Findings of our study do not support proposition P_2 (Table 2). The number of explicit assumptions is greater when there are no subcontractors involved.

P_3 is supported. Military clients have the maximum mean of explicit assumptions (Table 3).

Table 3. Findings related to p_3: Client type

p_3: Client type	Mean %	Mean	N	Std. Deviation
Military	29,94	2,3949	7	1,0362
International	16,41	1,3127	7	0,369
Civilian	22,92	1,8333	1	,
Total	21,71	1,7366	16	0,9858

Table 4. Findings related to p4: Current phase

p_4: Current phase	Mean %	Mean	N	Std. Deviation
Analysis	26,72	2,1375	4	0,6129
Not	20,04	1,603	12	1,0699
Total	21,71	1,7366	16	0,9858

P_4 is supported (Table 4).

Table 5. Findings related to p_5: RE tools utilization

p_5: RE tools utilization	Mean %	Mean	N	Std. Deviation
RE Tools Not Used	17,75	1,42	3	0,5188
RE Tools Used	22,62	1,8097	13	1,0672
Total	21,71	1,7366	16	0,9858

As depicted in Table 5, our findings support P_5.

Table 6. Findings related to p_6: RE training

p_6: RE training	Mean%	Mean	N	Std. Deviation
RE training not received	18,09	1,447	10	0,2501
RE training received	22,19	1,775	2	0,225
Total	18,77	1,5017	12	0,2116

Findings related to RE training support P_6.

3.3. 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 validity 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. The subjects were selected by the project managers at the companies, thus 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. This threat was alleviated to some degree by the fact that selected companies mostly had same CMMI levels.

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 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 [18, 19]. 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. We also have a small sample size.

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.

4. 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. The study focuses on the possible and not previously studied relationships between project characteristics and utilization of explicit assumptions by software engineers. We construct a base for future studies aiming to search for possible relationships between software project's characteristics and software engineers' behavior related to completing requirements by explicit assumptions. Due to a small sample size, we could not satisfy the hypotheses that we initially aimed for. Four out of six propositions are supported.

Improvements can be made regarding the means to measure parameters of the propositions. In addition to the man-month measure, line of code, cost, or other measures of project size may be used. As a dependent variable, the number of explicit assumptions divided by the number of requirements may also be used.

Factors impacting software engineers' preferences to fill gaps may not be limited to project-related specifications. Organization- and engineer- related factors, and interrelations between these factors may also be studied in future. Both companies in our sample are very similar with respect to organizational and software engineer-related parameters. 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 [3]. Further studies may also focus on incomplete quality-attributes-related requirements.

When the relationships between software engineers' preferences to complete deficient requirements and project-related parameters are identified, actions to prevent implicit assumptions may be taken.

6. References

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

[2] B. Boehm, "Identifying Quality-requirement Conflicts", *IEEE Software,* Mar 1996, Volume: 13, Issue: 2, pp. 25-35.

[3] B.H.C. Cheng, and J.M. Atlee, "Research Directions in Requirements Engineering", *Future of Software Engineering (FOSE '07), IEEE* 2007, 23-25 May 2007, pp. 285-303.

[4] B. Zagajsek, K. Separovic, Z. Car, "Requirements Management Process Model for Software Development Based on Legacy System Functionalities", *9th International Conference on Telecommunications*, (ConTel 2007), 13-15 June 2007 pp.115-122.

[5] C. Ebert, J. D. Man, "Requirements Uncertainty: Influencing Factors and Concrete Improvements", ICSE'05, Proceedings of International Conference on Sofware Engineering, May 15-21, 2005, pp. 553-560.

[6] D. Damian and J. Chisan, "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*, Volume 32, Issue 7, July 2006, pp.433-453.

[7] H.F. Hoffman, and F. Lehner, "Requirements Engineering as a Success Factor in Software Projects", *IEEE Software*, July-Aug 2001, pp. 58-66.

[8] H. Saiedian, and R. Dale, "Requirements Engineering: Making the Connection Between the Software Developer and Customer", *Information and Software Technology*, 42(2000), pp. 419-428.

[9] IEEE Std. 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE, 25 June 1998.

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

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

[12] J.K. Willoughby, "Adaptations to the Systems Engineering Management Process for Projects with Incomplete Requirements", *Proceedings of IEEE International Conference on Systems Engineering*, 24-26 Aug. 1989, pp.197-200.

[13] M. Agrawal, and K. Chari, "Software Effort, Quality, and Cycle Time:A Study of CMM Level 5 Projects", *IEEE Transactions on Software Engineering*, Volume 33, Issue 3, March 2007, pp.145-156.

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

[15] M. R. Strens , R. C. Sugden, 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, March 11-15, 1996.

[16] M. Schrage, "Never go to a client meeting without a prototype [software prototyping], *IEEE Software*, Volume 21, Issue 2, Mar-Apr 2004, pp.42-45.

[17] Lloyd, W.J., Rosson, M.B., and Arthur, J.D., "Effectiveness of Elicitation Techniques in Distributed Requirements Engineering", *Proceedings of IEEE Joint International Requirements Engineering Conference on Requirements Engineering*, 9-13 Sept. 2002, pp. 311-318.

[18] O. Albayrak, "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), pp. 68-87, IGI Global, 2009.

[19] O. Albayrak, M. Bicakci, H. Bozkurt, "A Study to Observe Relations Between Software Engineer's Responses to Incomplete Requirements and Requirements Volatility", *International Conference on Software Theory and Practice, SETP 2009*, July 13-16, 2009, Orlando, (accepted paper).

[21] R. B. Rowen, "Software project management under incomplete and ambiguous specifications", *IEEE Transactions on Engineering Management,* Volume 37, Issue 1, Feb. 1990, pp.10–21.

[22] R.P.D. Redondo, J.J.P. A.F. Arias, Martinez, B.B. Vilas, "Approximate Retrieval of Incomplete and Formal Specifications Applied to Vertical Reuse", *Proceedings of International Conference Software Maintenance*, 2002, 3-6 Oct. 2002, pp. 618-627.

[23] R.T. Yeh, and P. Zave, "Specifying Software Requirements", *Proceedings of the IEEE*, Sept. 1980, Volume: 68, Issue: 9, pp. 1077- 1085.

[24] S. Kujala, M. Kauppinen, L. Lehtona, and T. Kojo, "The Role of User Involvement in Requirements Quality and Project Success", *Proceedings of the 13th IEEE International Conference on Requirements Engineering RE(2005)*, 29 Aug.-2 Sept. 2005, pp. 75-84.

[25] U.V. Subramanian, U.V, "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), 7-10 Dec. 1999, pp. 188-195.

APPENDIX A:

Gap-seeded requirement used in the first section:

For the following software requirement, do one of the following 3 alternatives:
1. draw prototype screens for at least two inputs you enter,
2. write source code in any programming language you know (C/C#, Java...),
3. write pseudo code.

For any positive number entered by the user, the program should display a list of even numbers less than input.

PLEASE LIST ANY QUESTIONS/ASSUMPTIONS YOU HAVE FOR YOUR SOLUTION

APPENDIX B:

The types of gaps seeded in the requirement:

Gap Type	Related Assumption/Question
Input type	What is the type of input: Is it integer, double, float…?
Prompt	Which text messages are displayed to the user?
Order	What is the order of the list? Is it ascending or descending?
Format	What is the format of the output list? Appearance?
Application type	Is it a console, windows, or Web application?
Error messages	Which errors are displayed, and how to handle errors?
Stopping condition	What is the stopping condition while listing?
Validation	How is input validation realized?

APPENDIX C:

Project Characteristics:

Name/ Code	
Planned (man-month)	
Realized (man-month)	
Any subcontractor?	€ Yes € No
Client Type	€ Military € Civilian-State € Civilian-Private € International € Other (please explain)
Current Phase	€ Planning € Analysis € Design € Implementation € Test € Maintenance
RE tools used?	
RE training taken?	