2004 Command and Control Research and Technology Symposium
The Power of information Age Concepts and Technologies

An Empirical Assessment of the Impact of Requirements Uncertainty on Development Quality Performance

By

Ayad Y. Aldaijy, Ph.D

Royal Saudi Air Force, RSAF
Ministry of Defense and Aviation
P.O Box 102847
Riyadh 11685, Saudi Arabia
Tel: (9665) 324-5476
Fax: (9661) 478-7064
E-mail: ayad@aldaijy.com

An Empirical Assessment of the Impact of Requirements Uncertainty on Development Quality Performance

Ayad Y. Aldaijy, Ph.D
Royal Saudi Air Force, (RSAF)
Ministry of Defense and Aviation
P.O. Box 102847
Riyadh 11685, Saudi Arabia
Email: ayad@aldaijy.com

Abstract

System requirements are recognized as a critical step in the development of quality software (SW) systems and an important area of research. Being the first step in the process of software engineering, the effort has potential to shape the direction for all subsequent project activity. The main purpose of this research is to examine the impact of requirements uncertainty and task uncertainty on outcomes in software development projects, limiting the attention to process and product quality. Some of those examined projects are defense-related and aerospace-command and control systems. A cross-sectional survey of 123 participants work in software development in 34 U.S organizations was employed to prove my research model. Analyzed data provided evidence of a significant negative association between requirements uncertainty and development quality factors: process and product. Moreover, the analyzed data showed that there is a significance positive association between requirements and task uncertainty. In addition, the data provided evidence of a negative significant association between task uncertainty and process and product quality. My study pointed to areas where there was negative impact on the developed system quality. In particular, my research focused on the uncertainty regarding user requirements, because I believed that this had the most influenced. Findings from this research can provide the basis on which project managers and software practitioners can design concrete strategies that would enhance the performance of software development to high quality ends.

Keywords: Requirements uncertainty; task uncertainty; process quality; product quality.

1. Introduction.

The development of software project is a complex problem-solving task made difficult by the involvement of numerous customers and by a dynamic organizational environment in which information needs may change rapidly. The ultimate success of

a software project hinges on a clear and complete understanding of the problem to be solved as well as a thorough definition of the user's needs and expectations. This understanding is accomplished through a process known as requirements engineering (RE). Given the necessity of complete and accurate requirements for the development of a successful SW project, it is not surprising that requirements engineering is frequently and convincingly presented as the most critical phase of software development [1] [2] [3] [4] [5].

Despite continuous improvement during the past decades, controlling software quality remains the major challenge in software development projects. Many projects continue to experience problems or outright failures. Over the years, a set of tools and techniques, such as CASE tools, Rapid Application Development (RAD), information engineering, etc., have been undertaken. And yet, new products continue to fail to meet their functional, technical, and reliability objectives, often over budgets and late [6].

In fact, many reports have been published and outlined concerning software industry efforts. For example, the Chaos study, published by the Standish Group, indicates that 26 percent of software projects are successful, 46 percent are challenged, and 28 percent have failed [7]. Other findings, in the same study, show that the average cost overrun is 89 percent, the average schedule overrun is 122 percent, and 45 percent of the functions provided in newly developed systems are never used. Despite the costs, many reports suggest that project failures are occurring with alarming frequency. In 1996, annual U.S spending on software projects reached more than $270 billion, and 58 percent or $145 billion of this investment was a casualty of costs overruns and failed projects [7]. One explanation for the high failure rate is that mangers are not taking prudent measures to assess and manage uncertainty associated with the early stages of the software development.

Everyone is in agreement with the difficulty of defining requirements correctly and completely; but it is equally important that the developers are solving the right problem over time. Dean Leffingwell of Rationale software estimates that between 40 and 60 percent of software defects and failures can be attributed to requirements that are specified incorrectly [8]. However, a great deal of the problems in software development projects is due to uncertainty about user requirements and about the high dynamic complexity and ambiguity of the software development task. Uncertainty is broadly defined as the inability to specify something with precision. Uncertainty in user requirements, however, comes in a variety of forms, such as instability, diversity, and analyzability. These types of requirements often reflect on a poor understanding of the business processes, insufficient details for developers to do their tasks, and insufficient feedback from users over the life cycle of the software project.

Therefore, based on the discussion above and on a comprehensive relevant review of literature, the problem that I chose to address is:
There is a need to assess the software quality performance, with respect to RE, to identify the reasons for systems failure. It is important to establish a clear link between RE process and performance in order to provide a more integrated view of

the requirements activity and their relationships to the quality of software development projects. Moreover, the critical problem in today's practice has been a failure to understand the problem to be solved, as well as the real needs and requirements of users in order to build the right system.

In an analytical context and as the level of uncertainty around user-requirements increases, my research therefore developed and tested a theoretical model that demonstrated the perceived impact of requirements uncertainty (RU) and task uncertainty (TU) on outcomes in software projects, limiting the attention to the software process and product quality. Specifically, I followed following research process:

1. Examined the relationships between requirements uncertainty, task uncertainty, and software development quality factors;
2. Investigated to what extent the requirements uncertainty (RU) and task uncertainty (TU) issues shaped the performance of software development quality;
3. Assessed whether RU and TU are reliable indicators for predicting the quality of the development; and
4. Determined whether more practical methods could be applied in the RE process that espouse for coping with different aspects of requirements uncertainty (RU) and task uncertainty (TU).

2. Research model and hypotheses

2.1. Research Model

It is well understood that there are behavioral, economic and attitudinal outcomes associated with development projects. For reasons of scope, I limit the attention normally to quality outcomes. The issue of product and process quality is a success factor in most software projects. Product quality helps to ensure customer satisfaction and acceptance, proper operation in production, and reduced maintenance efforts over the shelf life of a system. One can assume then that the process of identifying the real needs plus accurate and complete requirements in a timely manner plays a role in ensuring product quality. Cooprider and Henderson [9] suggest that examining product and process outcomes together can reveal differential impacts of them on quality. Past research has identified some attributes of development quality, such as effective coordination [10], project completion within schedule and budget [11] [12], and overall quality of the development efforts [13]. Development process quality has been defined as the degree to which the process is designed to promote consensus among people participating in the development process, operate within established resources parameters, and reduce waste and redundancy [14]. Software product quality has been defined as the outcome of overall evaluation of the final product produced by the development process. Objective criteria have been considered, such as reliability and maintainability of the product, and subjective

criteria, such as user acceptance and satisfaction, as part of the overall assessment [14].

The requirements process management should have a direct impact on development quality, but it can also indirectly enhance these outcomes by controlling such uncertainties encountered with the process. There is little empirical evidence of the impact of requirements uncertainty on development quality. Markis et al., [15] stated that no studies have considered requirements in the perspective of overall product quality, e.g. usability and utility, nor have they attempted to link poor product quality to process failings. Therefore, my research examines whether process and product quality outcomes vary by the level of uncertainty encountered in the requirements of a development projects. Increased levels of requirements uncertainty add to task uncertainty, while increase in task uncertainty should negatively impact process and product quality. The examination of these direct relationships is complemented by an examination of the nature of the interaction between requirements uncertainty and task uncertainty. Figure 1 illustrates my research model derived directly from the literature, reflecting the relationships between requirements uncertainty, the two task factors, and the dependent measures two aspects of development quality.

2.2. Requirements uncertainty and quality oriented development outcomes

The concept of RU has been widely studied by information systems and SW researchers, partly because of the importance of identifying the users' requirements for SW development projects. Proper management of the requirements can have the single biggest impact on project performance, and frequent changes can create major problems. Unsatisfactory requirements can make it difficult to mange SW development process and to validate the software product; unfortunately, it is difficult elicit information concerning organizational values and beliefs during the RE process [16]. From an information processing viewpoint, RU refers to the difference in the information necessary to identify user requirements and the amount of information possessed by the SW practitioners [17] [18] [19]. Three important dimensions of uncertainty can be identified [20]:

1. Requirements instability: the extent of changes in user requirements over the course of the project.
2. Requirements diversity: the extent to which users differ among themselves in their requirements.
3. Requirements analyzability: the extent to which the process of converting user needs to a set of requirements specifications can be reduced to mechanical steps or objective procedures.

The above argument suggests the following hypotheses:

Research Hypothesis 1:
The degree of requirements uncertainty (RU) in a development project influences the degree of the two aspects of development quality:

Research Hypothesis 1.A: The degree of requirements uncertainty (RU) in development project is negatively related to the degree of process quality.
Research Hypothesis 1.B: The degree of requirements uncertainty (RU) in development project is negatively related to the degree of product quality.

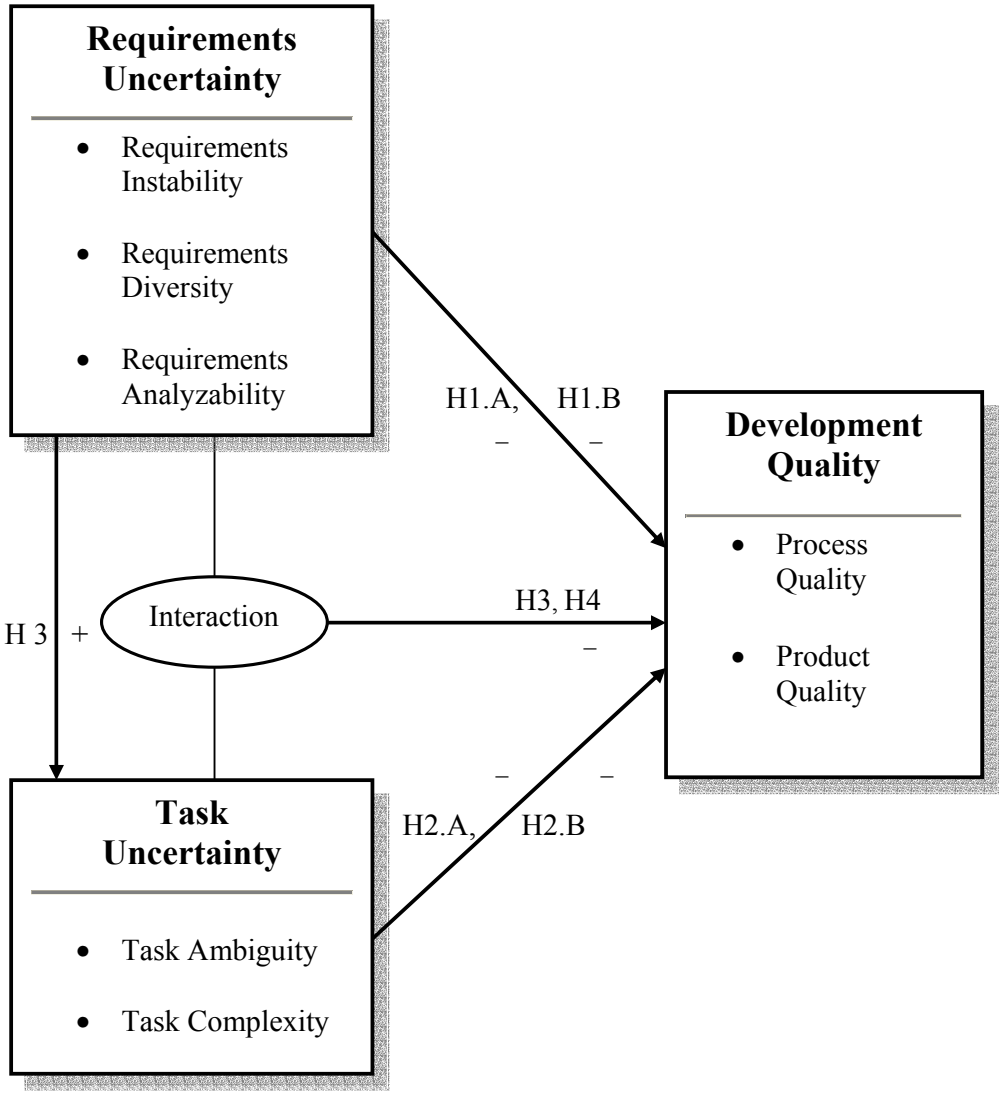


Figure 1: The Research Model

2.3. Task uncertainty and quality oriented development outcomes

Task uncertainty (TU) defined as "the degree to which work to be performed is difficult to understand and complex" [21]. In general, high levels of task uncertainty are associated with SW development [22] exacerbated by hard-to-predict factors in the development process. Ongoing changes contribute to the high level of uncertainty associated [23] [24]. Several studies support the theory that TU is a major determinant of information processing requirements. The concept of TU has been refined by other organization theory researchers along two dimensions [25] [26] [27], the first being task complexity (interdependence, autonomy, variety, structurability, intelligibility) and the second being ambiguity (predictability, controllability, exceptions, rate of change). Perceived complexity in task is widely acknowledged to be an important factor affecting the SW development process. Task complexity refers to the number of inputs, input variation, number of sub-tasks, and number of operations or procedures involved in the completion of a task [28]. For information processing or decision-making, a task that utilizes fewer information cues is considered as having lower task complexity than one with more cues. The higher the complexity, the more subtasks the decision-maker must complete [29].

Unlike task complexity, task ambiguity refers to those tasks for which multiple acceptable solutions exist, as perceived by those with different frames of reference [30]. Task information that is clear and directed leads to similar interpretations, while task information that is ambiguous leads to multiple interpretations that must be resolved in order to develop a shared understanding of how to perform the task. From an information processing perspective, the impact of task complexity has been widely documented, but less research has been conducted on the role of task ambiguity in the context of SW development [31]. Nor has there been any recent empirical work relating to influences of task complexity and ambiguity on the outcome of SW development quality.

Application development teams often include people with limited knowledge about the problem domain and detailed knowledge is often provided too late to help [32]. Inadequate information can result in decisions to delay certain steps or to execute the development process in a trial and error fashion [33]. Despite differences in definitions and theoretical approaches over time, the literature provides considerable evidence that increased task uncertainty leads to decreased development quality. This leads to the following hypotheses:

Research Hypothesis 2:
The degree of task uncertainty (TU) in a development project influences the degree of the two aspects of development quality:

Research Hypothesis 2.A: The degree of task uncertainty (TU) in development project is negatively related to the degree of process quality.

Research Hypothesis 2.B: The degree of task uncertainty (TU) in development project is negatively related to the degree of product quality.

2.4. Requirements uncertainty and task uncertainty

Everyone agrees that there is difficulty in defining requirements correctly and completely; but it is equally important that the software practitioners are solving the right problem over time. For example, users are tempted to treat requirements-specification as an unimportant exercise, so meeting on the 'right' solution over time becomes an extended exercise of trial-and-error; a system this is initially intended to support some clearly-defined business objectives may eventually meet none of them [34]. As discussed in the previous chapter, a great deal of the problems of software development projects is due to uncertainty about user requirements and the high dynamic complexity and ambiguity of the software development task [35]. Uncertainty encountered with requirements is broad based. It affects every aspect of the project development lifecycle. And it involves, to a large or small extent, every member of the development group, from the users to testers. When requirements are managed well and to certain standards, the requirements effort can greatly aid in the development process tasks; when uncertainties arise, deep and significant problems may occur. Thus, increased levels of requirements uncertainty add to task uncertainty and this leads to the following hypotheses:

Research Hypothesis 3:
The degree of requirements uncertainty (RU) in a development project is positively related to the degree of Task Uncertainty (TU).

2.5. Requirements uncertainty and task uncertainty interaction

The interaction relationships were proposed to show how RU influences the relationships between TU and development quality: the assumption being that development performance is dependent on the organization's ability to handle uncertainty through its information processing capability [36]. To achieve a maximum level of performance, cooperation must occur between an organization's information processing capability and the level of the uncertainty that it faces. This leads to the hypotheses:
Research Hypothesis 4: The interaction between requirements uncertainty and task uncertainty influences the process quality.
Research Hypothesis 5: The interaction between requirements uncertainty and task uncertainty influences the product quality.

3. The empirical study

3.1. Data collection and sampling

Data collection is the most important stage in the research design. Data were collected from employees who worked in software development, manage software projects, or deal with software quality issues to obtain opinions on software development and perceptions associated with process and product quality. These

types of employees had significant technical understanding of the software process and software product, had been involved in the project from start to end, and interacted with both upper and system management. This is consistent with the Huper and Powell's [37] recommendation that the person(s) most knowledgeable should be chosen as respondents. Data were collected through the research instrument, which was designed, pre-tested, and sent to six hundred and fifty senior IS executives located across the United States. The names of these executives and their companies were randomly selected from the *Directory of Top Computer Executives*. This random selection process was used, as I wanted to test the hypotheses using data about projects emanating from diverse organizational and industrial contexts.

A total of one hundred and fifty-seven of six hundred and fifty surveys were received. The survey data has been gathered from thirty-four organizations. This provided a response rate of approximately 24% of the total responses received. Fourteen organizations indicated that they did not develop software in-house. Twenty collected survey forms were discarded due to missing data or their being unusable. One hundred and twenty-three answered survey data forms were therefore used for the analysis. According to several researchers [38] and [14] stated that response rate of 24% is an acceptable average rate of response. Additionally, the sample size of 123 is not small compared to other studies of requirements or quality in software development. For example, the COCOMO model, based on one of the larger datasets, is estimated with the data on 63 software projects [39], [14] used 95, [20] used 64, [40] used 43, whereas [41] used 24 projects and [42] used 15. Furthermore, many of SW practitioners and managers who participated in the survey asked for an executive summary that I offer this if they wish. This is evident that these participants were interested in my research topic. The use of survey methodology in this research may raise the issue of bias, because the study required the respondents to reconstruct their project experience. Such recall problems were reduced, to some extent; by collecting data only on recently completed projects and ensuring that the quality scores were cross validated by a subset of software practitioners' responses.

I also tested to see if the sample was biased with respect to key characteristics, such as size of development team, respondent's positions, project duration; and project budget. Overall, the 123 projects showed a good dispersion of project context characteristics. In addition, a Multivariate Analysis Of Variance (MANOVA) was undertaken to determine whether differences in respondents in regard assessing the dependent variables, process quality, and product quality. MANOVA is a statistical technique used for assessing group differences across multiple metric dependent variables simultaneously, based on a set of categorical (non-metric) variables acting as independent variables. The participants were classified, according to their positions in their organizations, into four positions: software project manger, requirements engineer, software developer, and software engineer. The test indicated no significance differences by the respondent's positions (Wilks' Lambda = 0.93, F = 1.49, $p = 0.18 > 0.05$). Thus, while the response rate was low, the series of tests did not reveal any significant threats to the population of in-house developed software development projects. A profile of the projects is summarized in table 1, while table 2 provides a description of the study sample.

Table 1: Statistical profile of development projects

	Mean	Standard Deviation	Min	Max
Project Duration (months)	13.5	12.6	2	60
Project Budget ($000)	$24,600	$146,000	$35	$1.2 Billion
Full-time Emp. (#)	303	659	10	6,000
Persons-Months	49	167	2	1,200
Tech/Mgmt Hour	4,382	10,600	35	68,000

Table 2: Profile of respondents positions in their organizations

Respondent's Positions	Number of respondents	Percentage
Software Project Manager	38	31%
Requirements Engineer	35	28 %
Software Developer	32	26 %
Software Engineer	18	15%
Total	123	100%

Table 3: Respondents characteristics: Software project size

Software Project Size	Number of respondents	Percentage
Small (< 100 KSLOC)	44	34%
Medium to Large (>=100 KSLOC)	79	64%
Total	123	100%

KSLOC: Thousands of Source Line of Code

Table 4: Description of study sample

Industry	**Number of respondents**
SW and System Development	36
IT Services	24
Telecommunications	11
Environmental Protection	11
Insurance	10
Financial	10
Defense-related	7
Aerospace-Command & Control	6
Petroleum	5
Transportation	3
Total	123

3.2. Instrument development

A questionnaire (the survey instrument) was developed for the measurement and operationalization of the two independent variables (requirements uncertainty and task uncertainty) in the theoretical model as they directly or indirectly influenced the dependent variable (quality of the resulting software project). Items for specific constructs were drawn from established instruments. My research instrument, which is part of research effort on software development projects, was developed through an extensive review of the literature. The instrument consisted of three sections. The first addressed demographic data (individual's background and his/her organization). Items in the second section of the instrument related to the project characteristics, the development practices and techniques, and the nature of the development task. A series of 9 items involved multiple choice, Likert scale questions.

Furthermore, one question was included to assess the requirements engineering techniques used for the specified project and another open-end question to list by respondents the important contingencies that impacted the quality of their software development. The third section of the instrument was about the implementation outcome. This section listed items to measure the independent variables (requirements uncertainty and task uncertainty) and the dependent variable (development outcomes). A 5-point Likert-type scale that ranged from strongly disagree to strongly agree was used for each of the items of requirements uncertainty aspects (9 items). Another 5-point Likert-type scale that ranged from very small extent to very great extent was used for each of the items of task uncertainty aspects (5 items). The last 5-point Likert-type scale that ranged from very poor to very good

was used for each of the items of development quality aspects (9 items). All items in this section were derived from previous related studies.

A pretest stage was used to validate the questionnaire items derived from prior research. Subsequently, a pilot test was conducted using subject matter experts from the academia and industry. At the end of this stage, a number of modifications were made: 1) more background information on the project and company were added to the questionnaire; 2) some items were added; and 3) less important ones were deleted.

3.3. Measures for independent variables

I closely followed Straub's [43] suggestions for improving instrument validation. I used previously developed and adequately validated scales. Where necessary, I modified some of the words in the questions to suit the context of this study.

3.3.1. Requirements uncertainty

The measurements of the sources of requirements uncertainty (requirements instability, requirements diversity, and requirements analyzability) were assessed on a 5-point scale (1- Strongly Agree, 5 – Strongly Disagree), and derived from previous research paper [20]:

1. Requirements Instability: It refers to the extent of change in user requirements over the course of the project. It was measured by four items.
2. Requirements Diversity: This aspect refers to the extent to which users differ amongst themselves in their requirements. It was measured by three items.
3. Requirements Analyzability: This aspect refers to the extent to which a conversion process can be reduced to mechanical steps or objective procedures. In this study, the process of converting user needs to requirements specification was assessed by measurements of four items.

3.3.2. Task uncertainty

The variables used to measure the two sources of task uncertainty (task complexity and task ambiguity) are now described below. The items for each variable were assessed on a 5-point Lickert scale (1 - Very small extent, 5 - Very great extent).

1. Task Ambiguity: This variable was derived from previous research [31] and refers to the extent to which multiple acceptable solutions exist, as perceived by those with different frames of reference. It was measured by three items.
2. Task Complexity: This variable was derived from previous research [31] and refers to the degree to which work to be performed is difficult to understand and uncertain. It was defined in two factors: 1) The thinking time needed to solve the problem and (2) the complexity involved in the solution process to accomplish the task. It was measured by two items.

3.4. Measures for dependent variables

3.4.1 Process quality

This variable was used to measure the quality of development process. A 5-item scale was used to reflect some desirable characteristics of the process, such as the degree to which the project was completed on schedule, the degree to which the project met cost targets, and the degree of agreement among participants. The items used were adapted from [13], [44], and [11].

3.4.2 Product quality

This variable was used to assess the software product quality. A 5-item scale was used to measure SW product quality: reliability, flexibility, maintainability, system acceptance, and user satisfaction. The items used was adapted from [13], [44], and [45].

4. Results and discussion

4.1. Reliability Analysis

Reliability is defined as the consistency of a test (or measuring instrument) over time, across subjects, or within a test or scale [46]. There are four general procedures for determining the reliability of a test: stability, equivalence, combined stability and equivalence, and internal consistency [47]. Internal consistency measures the homogeneity of the items of an instrument; it is assessed after one administration of the instrument. This type of reliability typically uses the split-half method to avoid administering the same instrument twice to the same subjects [48]. There are number of procedures that determine the internal consistency of a test or a measuring instrument. However, I decided to employ the Cronbach's method of Alpha Reliability Coefficient because: (a) this method is generally the most appropriate for survey research and other questionnaire in which there is a range of possible answers for each item [47], (b) this method is commonly used for determining the internal consistency of the typical Lickert scale measuring instruments [48], and (c) the effect of this method is to produce a reliability coefficient that is approximately what one would obtain if one were to split the measuring items into all possible halves, calculate a split-half reliability for each split, and take the average of all the split-half reliability coefficients [48]. Using the Cronbach's Alpha method, the reliability coefficient of the variables of the questionnaire administered in this study was computed by a procedure in the Statistical Package Software (S.P.S.S). All coefficients were found to be statistically significant, indicating a high degree of reliability on the variables. Alpha values of 0.70 or above are acceptable indicators of internal consistency, as suggested by many researchers [49] [46]. Alpha values were calculated for each multi-item construct. As seen in Table 5.10, all the calculated alpha values were found to be above the 0.70 level, except for the scale for requirements instability (which had an alpha coefficient 0.65) and the requirements analyzability (which had an alpha coefficient 0.66). However, this scale was retained because: 1) it was very close to guideline; and 2) the correlation among their items

was significant even at level 0.001. Therefore, the data indicates the fact all the scales are reliable (table 2).

Table 5: Description of study sample

Variable	**Scale reliability**
Requirements Uncertainty:	.75
- Requirements Instability	.65
- Requirements Diversity	.77
- Requirements Analyzability	.66
Task Uncertainty:	.71
- Ambiguity	.74
- Complexity	.79
Development Quality:	
- Product Quality	.83
- Process Quality	.80

Table 6: Correlation matrix among aspect variables (N=123 software projects)

Variable	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Development Quality							
1. Product quality	1.0						
2. Process Quality	47	1.0					
Requirements Uncertainty							
3. Requirements Instability	-.42	-.37	1.0				
4. Requirements Diversity	-.20	-.29	.56	1.0			
5. Requirements Analyzability	-.35	-.36	.19	.17	1.0		
Task Uncertainty							
6. Task Ambiguity	-.22	-.29	.33	.34	.32	1.0	
7. Task Complexity	-.01	-.01	.10	.16	.10	.24	1.0

4.2. Inferential analysis of the data

Inferential analysis of the data was conducted to examine the research hypotheses derived from a review of the literature and it was based on the following statistical tests: (a) the simple correlation and (b) multiple linear regression analysis. Simple correlation was used for research hypothesis 1 (A and B), research hypothesis 2 (A and B), and research hypothesis 3. Multiple linear regression was used for research hypothesis 4 and research hypothesis 5.

Two independent variables were shown to have significant relationship with two aspects of software development quality: process and product. The findings were drawn from an examination of the research hypotheses. Analyzed data provided evidence of a significant negative association between requirements uncertainty and process quality ($r=-0.47$, $p<0.001$). The level of requirements uncertainty in a software project is a negative predictor of development quality; it explains about 22% ($r^2 =0.221$) of the variance of process quality. Requirements instability, an aspect of requirements uncertainty, appears more problematic than requirements diversity and requirements analyzability because it has a strong negative association ($r=-0.37$, $p<0.001$) with process quality in a software project. Additionally, analyzed data provided evidence of a significant negative association between requirements uncertainty and process quality ($r = -0.46$, $p<0.001$). The level of requirements uncertainty in the software project is also a negative predictor of development quality; it explains about 21% ($r^2 =0.211$) of the variance of product quality.

Moreover, the analyzed data showed that there is a significance positive association between requirements and task uncertainty ($r=0.40$, $p<0.001$), specifically, task ambiguity ($r=0.47$, $p<0.001$). The analyzed data did not however show statistically a significant positive relationship between requirements uncertainty and task complexity. May be software practitioners are more concerned by task ambiguity in software development projects rather task complexity.

In addition, the data provided evidence of a negative significant association between task uncertainty and process and product. The result indicated that the level of task uncertainty in a software project is a negative predictor of development quality; it explains about 5% of the variance of process quality ($r=-0.23$, $p<0.001$). However, surprisingly, task ambiguity, an aspect of task uncertainty, appears more important because it has the strongest negative association with process quality ($r=-0.29$, $p<0.001$) in a software project; it explains about 8% of the variance of process quality. Similarly, task ambiguity has shown a significance negative association with product quality ($r=-0.22$, $p<0.001$), it explains about 5% of the variance of product quality. It seems that task ambiguity has stronger impact on process quality than product quality. Additionally, task uncertainty had a negative association with product quality ($r=-0.17$, $p<0.001$) and less than task ambiguity; it explains about 3% of the variance of product quality. Further, task complexity did not show any relationships with the two aspects of development quality: process and product. Task complexity was considered as an unrelated variable. Task complexity may impact other software development phases and not necessarily the final phase. Based on this, my research

suggests that task ambiguity is distinct variable and must be kept separate from task complexity. Table 2 shows correlations between variables.

4.2.1. Interaction Relationships

The interaction relationships section pertains to the impact of the interaction between requirements uncertainty and task uncertainty on quality-oriented development outcomes beyond the main effects of the variable themselves. The first interaction pertains to the impact of requirements uncertainty on the relationship between task uncertainty and process quality; the second interaction pertains to the impact of requirements uncertainty on the relationship between task uncertainty and product quality as suggested by research hypothesis 4 and research hypothesis 5.
To test the interaction relationship of research hypothesis 4, two equations, with one multiplicative interaction and one without, were estimated. The results of the regression analysis for the interaction model and the main effect model for these variables: the interaction did not contribute to the process quality. These results reject research hypothesis 4 and accepts the null hypothesis. The lack of significance for the interaction term suggests that relating RU and process quality did not depend on the level of TU and vise versa.

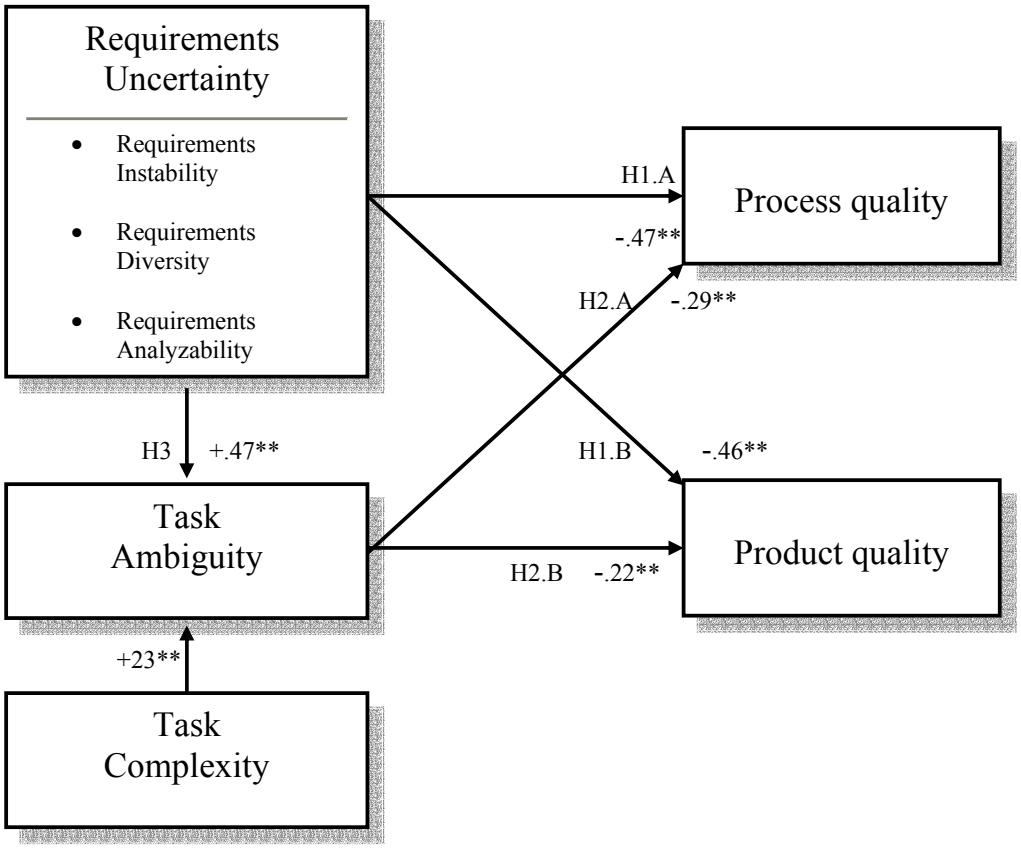
Furthermore, requirements uncertainty and task uncertainty do not interact with one another to influence the development outcomes. This means that requirements uncertainty and task uncertainty act independently of each other. Figure 2 shows the relationships among research model variables.

5. Conclusions and Future work

My study pointed to areas where there was negative impact on the developed system quality. In particular, my research focused on the uncertainty regarding user requirements, because I believed that this had the most influenced. The study presents practical data that points to ways to improve the development process and may lead to practices in which engineers and managers can shape the process to measure the quality of the process. The study also presents practical data that point out that uncertainty associated with software projects can easily imperil project success, threatening budgets, schedules, and down line integrity. An awareness of the sources of uncertainty and the means of reducing it should facilitate better planning and execution of software projects. The results of this research reveal conditions in the practice of requirements engineering that not only identify areas of weaknesses but also point to potential redress of the weaknesses.

While most of the previous requirements engineering research efforts were focused on software features through laboratory testing, there was little attention has previously been paid to measuring its effect in the field. My study showed that the effect measurements of requirements activity in the field do exist and can usefully contribute in the success of the software development projects if done properly. This

work is probably the first empirical attempt to examine the direct relationships between requirements activity and development quality performance. Findings from this research can provide the basis on which project managers and software practitioners can design concrete strategies that would enhance the performance of software development to high quality ends.



** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).

Figure 2: Relationships Found Among Research Variables

Future research might focus on ways on how to better plan and cope with the different aspects of requirements uncertainty. Future research can also overcome limitation of this study on the single respondent retrospective approach by capturing information from multiple sources internal and external to the project, and by employing a longitudinal data collection approach to better test causality between project characteristics and project outcomes. Additionally, a broader international

extension of this study should be conducted for future validation, since many software development projects are occurring globally. Studying the general state of requirements engineering in different countries and in different cultures, share their thinking on software projects issues and its management, would be very interesting research. Another valuable study would be a collection of methods for identifying and measuring quality with respect to requirements engineering, improvements in methods that predict project effort and duration based on qualities of requirements, and creative solutions to problems in the requirements gathering process. Studies such as these could provide industry with the data to continue its crucial examination of this central component to the software development projects.

5.1. Limitations of the study

Despite all the research advantages, this study is not without limitations: The study has at least two. One is that the data-collection approach that I employed relied on knowledgeable respondents whom were software project managers and engineers. These respondents assessed, after project completion, issues regarding the project as its initiation and its conclusion. Initially, I had sought to obtain a more balanced view by soliciting information from users, particularly regarding the development quality performance. However, considerable difficulty in collecting such information resulted in my abandoning this effort. The second limitation in this study is the response rate. The sample size of 123 is not small compared to other studies, but the sample size should be seen as large enough to draw some basic conclusions allowed by the tests, but a larger response base would have allowed for the exploration of issues in finer and more conclusive detail.

5.2 Operation and Policy Recommendation

Based on my research, it is possible to make a number of recommendations that should result in more effective ways of determining the real user needs and requirements.

1. Effective communication is a key factor in successful requirements engineering. It appears that effective communication occurs when managers and engineers are able to:
    - Identify and determine precisely and completely needs of their users and to map needs into system requirements and
    - Establish the right relationships with their users so that all parties can develop trust and agreements that facilitate coordination of their respective viewpoints.

    Therefore, multidisciplinary training for requirements practitioners is a matter of critical importance. The requirements engineers must posses both the social skills to interact with a variety of stakeholders, including potentially non-technical customers, and the technical skills to interact with system designers and developers.

2. There exists a general lack of knowledge regarding how much time should be allocated to the requirements engineering phase. Requirements engineering is often treated as a time-consuming, bureaucratic and contractual process. This attitude should be changed as requirements engineering is increasingly recognized as critically important.

3. Software project managers and engineers need to know that no two software development projects alike. As a result, there is no single requirement engineering technique that is applicable to all types of systems. Managers and engineers should select the technique that is appropriate to their software projects.

4. SW managers should regularly assess the external variables for each SW development environment and establish a log. At the same time, they may consider other technical and non-technical factors to be assessed in the project according to the development context. Therefore, the log data will show how the development projects are being managed. Additionally, this should improve the SW process and thus improve development quality performance.

5.3 Concluding Remarks

Requirements engineering is a process that presents many hazards to the success of software development projects. It is my hope that the findings of my research will help in providing better understanding of the requirements process in order to reduce uncertainties associated with the quality outcomes. In the meantime, I definitely agree with Osterweil and Clarke [50] who state that "the research into requirements and specifications should be sharply accelerated." I believe that there is a need to further explore the causes and effects of requirements related problems on the software development process.

Additionally, the demand for better, faster, and more usable software systems will continue, and requirements engineering will therefore continue to evolve in order to deal with different development environments. Indeed, I believe that effective requirements engineering will continue to play a key role in determining the successes or failure of projects, and in determining the quality of systems that are developed.

Bibliography

[1] T. Byrd, L. Cossick, & W. Zmud, A synthesis of research on requirements analysis and knowledge techniques, *MIS Quarterly*, 16(3), 1992, 117- 138.
[2] R. Bostrom, Successful application of communication techniques to improve the systems development process, *Information & Management*, 16 (4), 1989, 279-295.
[3] I. Vessey, S. Coger, Learning to specify information requirements: The relationship between application and methodology, *Journal of Management Information Systems*, 10 (2) 1993, 177-201
[4] J. Whitten, L. Bentley, and V. Barlow, *Systems Analysis and design methods* (Burr ridge: Irwin, 1998)
[5] G. Davis, Strategies for information requirements determination. *IBM Systems Journal*, 21(1) 1982, 4-30.
[6] J. Kuilboer, N. Ashrafi,. Software process and product improvement: an empirical assessment. *Information and software technology*, 42 (2) 2000, 27-34.
[7] Standish Group, The CHAOS Report, Web Report, (http://www.standishgroup.com/chaos.html, Standish Group International, Inc., 1998)
[8] B. Abott, Requirements set the mark. *Infoworld,* March 5, 2001, 48-50.
[9] J. Cooprider, J. Henderson, Technology processes fit: perspectives on achieving prototyping effectiveness, *Journal of management information systems,* 7 (3), 1991 67-87.
[10] R. Kraut, L. Streeter**,** Coordination in software development, *Communications of the ACM*, 38 (3), 1995, 69-81.
[11] M. Mahmood, System Development methods - A comparative investigation. *MIS Quarterly*, 11 (3), 1987, 293-311.
[12] C. Mcphee, A. Eberlin, *Requirements Engineering for Time-to-Market Projects*, Proceedings of the Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer Based Ayatems (ECBS'01), IEEE Computer Society, 2001, 252-260.
[13] M. Alavi, An Assessment of the prototyping Approach to Information System Development, *Communication of the ACM*, 27(6), 1984, 556-563.
[14] A. Rai, H. Hindi, The effects of development process modeling and task uncertainty on development quality performance. *Information & Management*, 37 (3), 2000, 335-346.
[15] P. Markis, A,. Sutcliffe, , & Economou, A Tracing Requirements Errors to Problems in the RE process, *Requirements Engineering*, 15(4), 1999, 134-151.
[16] D. Leefingwell, D. Widerig, *Managing Software Requirements A Unifies Approach*, (Addison-Wesley, Wesley Longman Inc, USA, 2000).
[17] R. Daft, N. Macintosh, A tentative Exploration into the amount and equivoality of information processing in organizational work units, *Administrative Science Quarterly*, 26 (2), 207-224.
[18] J. Galbraith, *Organizational Design* (Addison-Wesley, Reading, MA, 1977).
[19] M. Tushman, and D. Nadler, Information Processing as an integrating concept in organizational design, *Academy of Management Review*, 3(5), 1987, 613-624.
[20] S. Nidumolu,. The effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening variable, *Information Systems Research*, 6(3), 1995, 191-219.

[21] J. Alexander, W. Randlph, The fit between technology and structure as a predictor performance in nursing subunits, *Academy of Management Journal*, 28(4), 1985, 844-959.
[22] H. Krasner, B. Curtis, N. Iscoe, *Communication Breakdowns and Boundary Spanning Activities on Large programming Projects,* Proceedings of Empirical studies of Programmers: Second Workshop, Washington D.C. USA, 1987, 47-64.
[23] J. Cooper, Software Development management planning, *IEEE Transactions on Software Engineering* 10(1), 1984, 22-26.
[24] W. Humphrey, B. Curtis, Comments on a critical Look. IEEE Software 8 (4), 1991, 42-46.
[25] J. Dutton and J. Webester, Patterns of interest around issues: The role of Uncertainty and feasibility, *Academy of Management Journal*, 31 (3), 1988, 663-675.
[26] C. Gresov, R. Drazin, & A. Van, Work-unit task uncertainty design and Morale. *Organization Studies*, 10 [1], 1989, D45-D62.
[27] F. Milliken, Three types perceived uncertainty about the environment: State, effect and response uncertainty, *Academy of Management Review*, 12(1),1987, 133-143.
[28] J. Spence, R. Tsai, On human cognition and the design of information systems, *Information & Management*, 1997 (32), 65-73.
[29] J. Spence, R. Tsai, On human cognition and the design of information systems, *Information & Management*, (32), 1997, 65-73.
[30] L. Argote, Input uncertainty and organizational coordination in hospital emergency units, *Administrative science Quarterly*, 27(3), 1982, 420-434.
[31] S. Sussman, P. Guinsan, Antidotes for high complexity and ambiguity in software development, *Information & Management*, (36), 23-35.
[32] D. Parnas, P. Clements, A rational Design process: how and why to fake it, *IEEE transactions on Software engineering*, 12 (2), 19 86, 251-257.
[33] B. DeBrabander, G. Thiers, Successful information Systems development in relation to situational factors which effect effective communication between MIS-users and EDP-specialists, *Management Science* 30 (2), 1984, 365-375.
[34] F. Redmill, *Software Projects: Evolutionary versus Big-Bang Delivery* (Wiley Publications, Chichester, U.K 1997).
[35] A. Davis, *Software Requirements, Objects, Functions, and State (*Prentice-Hall publications, Englewood Cliffs, N.J., USA, 1993)
[36] M. Tushman, D. Nadler, Information Processing as an integrating concept in organizational design, *Academy of Management Review*, 3, 1987, 613-624.
[37] D. Huper, D., Powell, Retrospective reports of strategic level managers: guidelines for increasing their accuracy, *Strategic Management*, 6(2), 1985, 171-180.
[38] W. Kettinger, V. Grover, The Use of Computer-Mediated Communication in an Inter-Organizational Context, *Decision Sciences*, 28(3), 1997, 112-120.
[39] B. Boehm, *Software engineering Economics* (Prentice-Hall, Inc., NJ, USA, 1981)
[40] M. Kerishnan, C. Kriebel, K. Sunder, & T. Mukhopadphyay, An empirical Analysis of productivity and quality in software products. *Management Science*, 46(6) 2000, 189-197.
[41] A. Albrecht, J. Gaffney, Software function, source lines of code, and development effort prediction: A software science validation, *IEEE transactions on Software Engineering*, 9(6), 1983, 639-647.

[42] C. Kemerer, An empirical validation of software cost estimation models, *Communications of the ACM*, 30(5), 1987, 416-429.
[43] D. Straub, Validating instruments in MIS research, *MIS Quarterly*, 13(2) 1989, 147-169.
[44] J. Henderson, S. Lee, Managing I/S design teams: a control theories perspective, *Management Science*, 38(6), 1992, 316-327.
[45] J. Shin, A. Lee, A process model of application software package acquisition and implementation, Journal *of Systems and Software*, 32 (1), 1996, 57-64.
[46] R. Slavin, *Research methods, second edition*(Boston, MA, USA: Allyn and Bacon, 1992)
[47] S. Schumacher, J. McMillan, *Research in education: A conceptual introduction* (New York: Harper Collins College Publishers (3rd edition), USA, 1993).
[48] T. Crowl, *Fundamentals of educational research*, (Dubuque, IA: Wm. C. Brown Communications, USA, 1993).
[49] L. Gay, P. Diehl, *Research Methods for Business and Management*,
(New York: Macmillan Publishing Company, USA, 1992).
[50] L. Osterweil, L. Clarke, (1999), A proposed testing and analysis Research initiatives, *IEEE Software*, 23 (9), 1999, 89-98.