SOFTWARE SUBCONTRACTING SUCCESS:

A CASE STUDY ON THE RELATIONSHIP BETWEEN PROJECT SUCCESS AND PROCESS METRICS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF INFORMATICS

OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

KEREM YÜCETÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF INFORMATION SYSTEMS

NOVEMBER 2005

Approval of the Graduate School of Informatics

_____

Assoc. Prof. Dr. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Onur Demirörs
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Semih Bilgen
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Onur Demirörs	(METU, II)	_____
Prof. Dr. Semih Bilgen	(METU, EEE)	_____
Levent Alkışlar (MSc.)	(Aselsan Inc.)	_____
Dr. Çiğdem Gencel	(METU, II)	_____
Dr. Altan Koçyiğit	(METU, II)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Kerem Yücetürk

Signature:

ABSTRACT

SOFTWARE SUBCONTRACTING SUCCESS:
A CASE STUDY ON THE RELATIONSHIP BETWEEN PROJECT SUCCESS AND PROCESS METRICS

Yücetürk, Kerem

MSc., Department of Information Systems

Supervisor: Prof. Dr. Semih Bilgen

November 2005, 69 pages

While software subcontracting is a common business transaction in the information technology field, metrics specifically aimed at effectiveness of software subcontracting arrangements are not commonly available. This thesis makes a review of the literature and derives such metrics from fields of software quality, COTS acquisition and IS success. A case study is performed on software subcontracting projects of a Turkish defense contractor, and the project metrics are compared according to their success. The results suggest that metrics regarding the requirements are good indicators for subcontracting success and that larger projects may enjoy more success compared to smaller ones.

Keywords: Software Subcontracting, Software Quality Metrics, Software Project Success, Software Subcontracting Case Study

ÖZ

YAZILIM ALTYÜKLENİCİ İLİŞKİLERİNDE BAŞARININ METRİKLERLE İLİŞKİSİ KONUSUNDA ÖRNEK OLAY İNCELEMESİ

Yücetürk, Kerem

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Prof. Dr. Semih Bilgen

Kasım 2005, 69 sayfa

Bilişim Teknolojileri alanında yazılım altyükenici kullanımı yaygın bir uygulama olmasına rağmen, bu alanda etkinliği saptamaya yarayacak ölçütlerin neler olabileceği konusunda bilgi yaygın olarak bulunmamaktadır. Bu tezde literatürdeki yazılım kalitesi, hazır yazılım edinme ve bilişim sistemlerinde başarı alanları incelenerek, bu alanlardan yazılım altyüklenici etkinliğini belirleyecek ölçütler çıkarılmaya çalışılmıştır. Türkiye savunma sanayiindeki bir firmanın yazılım altyüklenici vakaları incelenerek sözkonusu projeler arasında başarılı olma durumlarına göre bir karşılaştırma yapılmıştır. Yapılan çalışma yazılım gereksinimi ölçütlerinin altyüklenicilere yaptırılan işlerin başarısını yansıtmada uygun olabileceğini ve büyük projelerin küçük projelere göre daha başarılı olabildiğini göstermiştir.

Anahtar Kelimeler: Yazılım Altyüklenici Yönetimi, Yazılım Kalite Ölçütleri, Yazılım Projelerinde Başarı, Yazılım Altyüklenicisi Vaka Çalışması

ACKNOWLEDGMENTS

I would like to thank Prof. Dr. Semih Bilgen, for being an impeccable guide, presenting me with possibilities, teaching me the right way to do things, encouraging me while I was writing this thesis. I cannot imagine writing a thesis without his guidance.

I am grateful to Aselsan, for enabling me to pursue a master's degree while working and allowing me to use company data for my thesis.

My colleagues at Aselsan, who helped to answer numerous questions, went over the numbers and documents for me, made time for discussing projects have been most helpful. Without them, this thesis would not be possible. Thank you.

My family supported me throughout the research and the writing as they have done for all my life. I thank them for always being there and wanting the best for me.

Thank you to Gamze, for the moral support throughout the months I've been writing. Without her support and care, I wouldn't have been able to complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

AQAP:	Allied Quality Assurance Publications
CAPM:	Contractor Acquisition and Performance Monitoring Process for Software Contracts
COTS:	Component Off The Shelf
DNA:	Data Not Available
IEEE:	Institute of Electrical and Electronics Engineers
IS:	Information System
ISO:	International Organization for Standardization
IT:	Information Technology
KSLOC:	Thousand Source Lines of Code
MASS:	Method for Assessing Software Subcontractors
MST:	Microwave and System Technologies (Turkish, Mikrodalga ve Sistem Teknolojileri)
NASA:	National Aeronautics and Space Administration
NATO:	North Atlantic Treaty Organization
SA-CMM:	Capability Maturity Model for Software Acquisition
SLOC:	Source Lines of Code
SW-CMM:	Software Capability Maturity Model
RFP:	Request For Proposals

UKSMA: United Kingdom Software Metrics Association

YMM: Software Engineering Department (Turkish, Yazılım Mühendisliği Müdürlüğü)

CHAPTER 1

INTRODUCTION

Commercial software development is a complex task that requires skilled developers and time. Companies short of capable developers or lacking expertise in certain development areas turn to other software companies for providing the development services. Software subcontracting is undertaking of the whole or parts of a software development project by a company under the guidance of the prime contractor for the software.

Software subcontracting involves complex relations between the prime contractor and the subcontractor. The requirements of the software have to be clearly communicated to the subcontractor. Any confusion on what is required in the resulting product may result in an unsatisfactory outcome for both the prime contractor and the subcontractor.

According to the necessities of the tasks in subcontracting agreement, a price has to be paid by the prime contractor, or the customer. A contract is drawn up, involving legal conditions that protect the participants in case of conflicts among the parties.

The development phase of a software subcontracting arrangement may require involvement of the customer about design, interfaces with other software, detailed requirement analysis. This is usually described as the contract management phase for subcontracting agreements. During this phase, the customer has to be involved to some extent to check if the product is shaping according to the needs defined by the customer.

Once the development efforts are over, there is usually a maintenance period in software subcontracting projects. During this phase the subcontractor pledges to fix any problems observed during the use of the software by the end user and make minor additions when necessary.

For each of the subcontractor selection, contract management and maintenance phases of software subcontracting, involvement and close cooperation of stakeholders is recommended by the major software acquisition models in the literature. For example, Software Acquisition Capability Maturity Model, states involvement of the management of the acquiring organization necessary to achieve higher levels of maturity (SEI, 2002). The acquirer has to be involved with the subcontractor to manage requirements, track subcontractor performance and evaluate the resulting product. This study, however, investigates the possibility of controlling subcontracting activities through metrics, without necessarily looking into the relatively long term undertaking of improvement of software subcontracting processes.

Measuring software development processes and the resulting software products becomes increasingly common in the software industry to gain insight into the effectiveness of processes and costs of the development efforts. Through analysis of the collected metrics, the status of projects can be tracked quantitatively, the development processes can be improved and estimations for future projects can be based on solid data. The importance of metrics and measurement of processes for software development is also underlined in the Capability Maturity Model of Software Engineering Institute (Paulk et al., 1993) and ISO 15504 (1998), the major frameworks for assessment of software processes.

Metrics can also be used in the software subcontracting arrangements to estimate the resources required for a development project and to track the progress of a development project. Through use of metrics in software subcontracting arrangements, outcomes of development efforts can be better controlled.

1.1 Thesis Objective and Scope

The objective of this study is to explore organizations' possibility of controlling software subcontracting effectiveness through use of metrics. In order to examine this, a literature survey was conducted about the factors that determine software to be effective. The metrics pertaining to these factors were collected.

The author of this thesis works for Aselsan Inc, a major defense contractor in Turkey. In order to obtain data about actual software subcontracting cases, some of the software subcontracting cases from the two Software Engineering Departments (YMM) of Aselsan's MST division were examined and their metrics collected. YMM is the acronym in Turkish for Software Engineering Department and will be used throughout the text.

Three software subcontracting agreements were examined involving a total of fifteen software development projects. The data was collected by examining existing metrics and documents as well as conducting interviews with YMM personnel working on the projects. However, exact data for only four of the fifteen projects were available, thus comparison was only possible among these four projects.

Success indicators for software subcontracting were established through a survey of the literature on the subject and interviews with Software Project Team Leaders in YMM. The metrics and success indicators about the cases under study were collected and relationships between metrics and success indicators were sought.

The aim of this thesis was not to arrive at a solid theory about a software subcontracting framework using metrics, but to look for relations between the metrics and success indicators that could lead to formulation of research questions for further exploration on the subject.

As noted above, rather than an extensive undertaking towards improvement of the whole subcontracting process, the scope of this study has been restricted to the investigation of whether it is at all possible to control via objective measurements, the level of success achievable in subcontracting arrangements.

1.2 Thesis Outline

The thesis is organized into five chapters. The second chapter presents a literature survey on the subject of software subcontracting, software quality, subcontractor selection and information system success models.

The third chapter describes the case study methodology used in this thesis. It then presents the suggested metrics for controlling software subcontracting effectiveness. Success indicators, quantitative values that allow for a comparison of the software subcontracting success of different projects are specified. Finally, the application of the metrics to YMM subcontracting projects is presented, along with further contextual data about the projects obtained through interviews.

The fourth chapter examines the results and looks for relationships between the metrics and the success indicators. The fifth chapter presents the conclusion of the thesis along with limitations of the work and suggestions for further study.

CHAPTER 2

LITERATURE SURVEY

The aim of this thesis is to investigate the relations between software subcontracting effectiveness and subcontracting process metrics. This presents a challenge, as it was not possible for this author to find a single work in the literature that focuses on providing metrics for software subcontracting. There is, however, work in the literature about software quality, subcontractor selection processes of firms and information system success models. Metrics for software subcontracting can be derived using this work from the fields of information systems and computer science.

Below, these well established research areas are briefly reviewed from the viewpoint of determining metrics relevant to software subcontracting success.

2.1 Software Quality – ISO 9126

ISO 9126 is a standard established by the International Organization for Standardization regarding software quality. It defines three technical reports: one

for Internal Metrics, one for External Metrics and finally one for “Quality in Use” Metrics. Internal metrics consist of metrics regarding the software itself such as code complexity value for functions. External metrics are metrics that are derived from the behavior of the software such as conformance to requirements or number of defects. Quality in use metrics try to capture the effects of using the software in a specific context (ISO, 2003). The standard suggests metrics that can be used to measure software quality, depending on the circumstances of a given software product. The standard states that the metrics that are suggested by the standard can be modified to suit individual project or organizational requirements for software quality. The standard also states that, the intended users of the standard include “developers, acquirers and independent evaluators” for “software product evaluation”.

The internal metrics technical document (ISO 9126-3) states metrics that can be applied to “request for proposal, requirements definition, design specification or source code”. These can be collected while the project is in progress and can be used to track the project status and quality of intermediate products. They also provide a prediction for the quality of the final product.

The external metrics technical document (ISO 9126-2) presents metrics that try to measure the quality of the software as part of the system that it operates in. These metrics can only be evaluated once a product is complete and in testing stage or in actual operation. The measurements for these metrics take place in the environment where the software is specified to operate.

The quality in use metrics technical document (ISO 9126-4) aims to capture the quality of the software through how much it achieves its goals in terms of "effectiveness, productivity, safety, and satisfaction in a specified context of use".

2.2 COTS Acquisition / Selection

Acquisition of COTS software can be regarded as somewhat similar to software subcontracting as in both cases software development efforts take place outside the acquiring company. Subcontracted software is more customizable from the acquiring company's perspective; therefore including metrics regarding conformance to the system and user requirements of subcontracted software also seem appropriate to measure its effectiveness. Still there are some metrics that can be borrowed from measuring COTS effectiveness for evaluation of software subcontracting.

Torchiano et al. (2002) claim that the set of characteristics or attributes they have compiled together to select COTS software have similarities with ISO 9126. They use ISO 9126 in a master's course they devise for classifying COTS software, but state that their study is also valuable in an industrial setting. They also suggest some further metrics for COTS software evaluation where they claim ISO 9126 falls short.

According to Torchiano (2002), well defined requirements lead to a very strict selection process for COTS, while vague and/or undefined requirements allow an easier selection process, as the requirements can be shaped according to the COTS product that is chosen. In this aspect, COTS selection and software subcontracting differ, as in software subcontracting the best results are achieved through well documented and stable requirements.

75% of the attributes that are used to classify and characterize COTS software products by Torchiano are qualitative. The authors argue that, "Instead of decomposing the [qualitative attributes] into atomic attributes, [it was] preferred to keep them as they were and to ask for qualitative values [as these] are more valuable in the educational context where the attributes were defined."

Torchiano et al. (2002) do not include internal characteristics that depend on source code for their evaluation of COTS products. They also include some other attributes such as maturity in marketplace, market share, software requirements, hardware requirements, product support, license type, acquisition cost, and other "domain specific" attributes.

Bertoa et al. (2002) have also defined a set of evaluation metrics for COTS components that are based on the ISO 9126 model. The authors claim that deriving a set of metrics is necessary because the international standards (they specifically name the IEEE and ISO standards) are "too general for dealing with the specific characteristics of software components." They provide 30 metrics to be used for

assessing COTS software. The metrics are designed to give clues about the suitability, interoperability, compliance, compatibility, maturity, learnability, understandability, operability, complexity, changeability, testability.

2.3 Subcontractor Selection

In this section, commonly accepted or recommended practices for software subcontractor selection are reviewed.

According to Assman and Punter (2004), there are three phases for software subcontracting: looking for subcontractors (selection), contract management, completion of contract (maintenance). So for selection of metrics for evaluating software subcontracting, different types of metrics for each of the phases could be used. Assman and Punter assess only the first phase (subcontractor selection) of subcontracting in their paper. They compare the existing standards and their associated guidelines for subcontractor selection and present their own methodology MASS: Method for Assessing Software Subcontractors. They compare SW-CMM, SA-CMM, ISO 9000, ISO 15504, Bootstrap, EuroMethod, IEEE Recommended Practice for Software Acquisition, and the Software Program Managers Network Model and define a method based on the best features of these standards.

2.4 Software Production Process

Software engineering has traditionally focused on a wide spectrum of software metrics. In this section, the literature on software production process metrics shall be reviewed from the standpoint of the current study.

Regarding the process of producing the software, Hyatt and Rosenberg (1996) describe the metrics that are used for software quality at NASA Goddard Space Flight Center. They identify the risks to the successful completion of a software project in the correctness, reliability, maintainability, reusability, schedule areas and identify metrics to measure whether the occurrence of a risk is imminent.

In accordance with ISO 9126, Hyatt and Rosenberg derive four goals to guide the metrics that they collect. These are Requirements Quality, Product Quality, Implementation Effectiveness and Testing Effectiveness. The metrics collected with these goals in mind are applicable even in the early phases of software projects, which is critical for NASA to identify potential problems early.

The requirements quality category of Hyatt and Rosenberg (1996) compiles metrics such as “number of weak phrases”, “number of optional phrases”, the ratio of changes to total number of requirements, number of untraceable requirements. Through these metrics, they decide on the ambiguity, completeness, understandability, volatility and traceability of requirements.

Product (or code) quality metrics includes complexity of the code, size of code, correlation of complexity and size, comment percentage. Through these metrics, they are able to understand the structure/architecture, maintainability, reusability and internal and external documentation attributes regarding product quality.

Implementation effectiveness metrics are about the management side of the project with metrics for the staff hours spent, and planned vs. actual task completions.

Finally, Hyatt and Rosenberg's (1996) testing effectiveness metrics have to do with the correctness of the resulting product, with number of errors and their criticality, time of finding the errors, time of fixing the errors, and the location of the errors.

According to the Contractor Acquisition and Performance Monitoring Process for Software Contracts (CAPM) of Software Engineering Process Office of United States Navy (2000), there are four goals of software subcontracting which are derived from SW-CMM:

1. Selection of the subcontractor
2. Agreement upon the "software standards, procedures, and product requirements" between the acquirer and the subcontractor.
3. Maintenance of technical and administrative communication throughout the duration of the contract.
4. Tracking of subcontractor's performance against requirements and using the data to reduce risk.

CAPM (2000) dictates that appropriate preparation needs to be performed by the acquirer in order to succeed in the subcontracting management operations. These include documentation, making the necessary resources available, and training the personnel with the necessary skills. Therefore any measurement of software subcontracting process effectiveness should also include metrics determining the preparation level of the acquiring party.

Li and Smidts (2003), have conducted a survey among experts in the software engineering field to find best software reliability indicators. They asked a panel of experts to rank 30 of 78 software engineering measures selected from an earlier study by Lawrance et al. (1998). Rankings of measures for Requirement, Design, Implementation and Testing phases of software development were identified using statistical methods.

For the requirements phase, “Fault Density”, “Requirement Specification Change Requests” and “Error Distribution” were found to be top indicators for reliability of the software resulting from the requirements. In the design phase, the top indicators were: “Design defect density”, “Fault density” and “Cyclomatic complexity”. For the implementation phase, “Code defect density”, “Design defect density” and “Cyclomatic complexity” were identified as indicators. Finally, for the testing phase, “Failure rate”, “Code defect density” and “Mean time to failure” were described as top reliability indicators.

Reliability of software can be seen as a source of software quality. Since development phases are carried out by the subcontractor in software subcontracting, in order to ensure the resulting software has satisfactory quality, the acquirer of the software could use the indicators described by Li and Smidts (2003) as metrics during the contract management phase.

2.5 Information Systems Success Models

Since software lies at the core of information systems, it is appropriate to look at software subcontracting from an IS success point of view. Looking at models for IS success in the literature, there are product based and process based models. Of these, the product based models seem appropriate for evaluating the effectiveness of subcontracted software as the result of a subcontracting agreement is usually a piece of software product. On the other hand, the development process of subcontracted software involves an intricate set of relations among the developers in the subcontracting company, and another set of relations with their clients in the acquiring organization. Therefore, the process based models also have a contribution to make for evaluating software subcontracting effectiveness.

The main issue in which information systems success differs from subcontracting effectiveness is that, IS success tries to measure how an entire software based system has made business processes of an organization more effective, while software subcontracting is usually only limited to effective selection of subcontractors for a software product and effective management of the contract

during product development and maintenance. Using this perspective, subcontracting could be considered as a subset of the activities that end up creating an IS system.

According to Drury and Farhoomand's IS success model (1998), an IS system's characteristics are formed by its storage, cost, processing and communication parameters. These characteristics, in turn determine user requirements, quality attributes, and outcomes or results such as the reduction in paperwork or increase in efficiency caused by the implementation of the IS system. The requirements, quality attributes and the outcomes determine the success or failure of an IS system.

The most cited work about IS success in literature belongs to DeLone and McLean. Their landmark paper (DeLone and McLean, 1992) described the information system success as being a product of system quality and information quality which in turn determine the use and user satisfaction of the IS, which in turn lead to individual impact for the user and further to organizational impact for the organization deploying the information system. In their 2003 update to the 1992 paper, the authors also factor in service quality alongside information and system quality (DeLone and McLean, 2003). They also combine the impact to the user and the organization under net benefits, using this term to encompass the benefits of an IS system that may be further reaching than the original recipients of IS benefits.

Myers, Kappelman and Prybutok (1997), state similar IS success dimensions to DeLone and McLean (1992), and present measures for assessing the IS success with

the aim of providing a comprehensive IS assessment system. In order to arrive at the measures, the authors have scanned the literature for measures that have been suggested and combined the results to arrive at a comprehensive assessment model.

Seddon (1997) disagrees with DeLone and McLean's model regarding IS use (1992). He states that while IS use is a prerequisite of IS success, it is not a factor. Seddon maintains that the user satisfaction from an IS system leads to expectations about the net benefits of IS use and later to IS use, therefore it is a consequence of IS success.

Considering these IS success parameters through a software subcontracting point of view, metrics that pertain to the requirements of a software, as well as metrics derived from quality attributes and usability attributes, as discussed earlier in section 2.1, can be used to determine the effectiveness of a software that is at the heart of an IS system.

CHAPTER 3

NATURE OF THE CASE STUDY

3.1 Case Study Methodology

The aim of this thesis was to understand how software subcontracting was being implemented in YMM, and investigating the relation between subcontracting success and a set of metrics related to the subcontract process. If such relations were thoroughly and consistently established, they could be used for enabling better management of the contracts. The metrics were derived from the literature, and their applicability was examined through a case study of the current subcontracting arrangements in YMM.

In order to understand the nature of software subcontracting arrangements in YMM, a qualitative case study approach was employed. Although the presence of metrics may suggest quantitative research methods, quantitative research requires large sample sizes. (Benbasat et al., 1987) When trying to solve real world problems, such a sampling is usually not possible. The sample size used in this study involved three contracts involving fifteen software development subprojects. Although

fifteen is a fairly large number, metrics for only four software development subprojects was found to be available. Due to this reduced sample size, the use of qualitative research methods was preferred. Furthermore, the main aim of this study is to formulate a hypothesis about the relationship between software subcontracting success and the metrics related to the outsourcing process. As such, qualitative research is considered to be a viable approach.

Kaplan et al. (1988) point out that it is usually necessary to combine qualitative and quantitative methods to understand the context-specific and context-independent aspects of a situation. Quantifiable measures of a situation, while valuable, are almost never possible to obtain as social systems involve numerous "uncontrolled – and unidentified – variables". Trying to obtain objective and testable results may come at the cost of being unable to gain a deeper understanding of the phenomenon.

According to Benbasat (Benbasat et al., 1987), "a case study examines a phenomenon in its natural setting" while using different sources of information without any manipulation or control of the environment. He also states that "case studies are more suitable for exploration, classification and hypothesis development" and that the results derived from the case study may depend on the investigator.

Similarly, Perry et al. (2004) state that "case studies are well suited to "how" and "why" questions in settings where the researcher does not have control over variables and there is a focus on contemporary events".

The case study described in this thesis matches the four criteria for case studies defined by Benbasat et al. (1987):

a. software subcontracting practices in YMM are examined in their natural setting
b. the study focuses on contemporary subcontracting arrangements
c. there is no control or manipulation of the subcontracting activities
d. there is no established theoretical base for software subcontract management using metrics

Therefore, it may be concluded that the choice of a case study was appropriate for the subject of software subcontracting control framework using metrics.

According to Benbasat et al. (1987), in the information systems field, the practitioners are usually ahead of the research. Therefore, in order to understand the IS field, the researchers must study the innovations fashioned by the practitioners.

The unit of analysis for this case study is the subcontracting projects in Aselsan's MST division's YMM departments. The projects for the case study were chosen due to their recentness and having a large body of data in the form of defect reports and documentation. Also, it was possible to conduct interviews with the YMM personnel involved in the subcontracting activities due to the currency of the subcontracting activities. As described below in section 3.4.1, the major methods of data collection applied in this study were interviews with selected YMM personnel and investigation of available documentation.

The case study results will hopefully be valuable due to the insight they provide about the context and details of the software subcontracting agreements of a company that develops and acquires outsourced software in the Turkish defense industry. It must be stressed here that the thesis is a case study of the software subcontracting activities in YMM. It does not aim to test a theory about software subcontracting. The main objective is to investigate software subcontracting phenomena, and question the possibility of connections among the measured metrics and the subcontract contexts with the success of the given subcontracting arrangements.

3.2 Subcontracting Metrics

In this section, metrics relevant to the software development subcontracting within the context of YMM shall be determined.

This thesis aims to perform a case study through evaluating subcontracting arrangements of the two software engineering departments of Microwave and System Technologies (MST) division of Aselsan Inc. While the nature of subcontracting projects changes in each case, essentially the software products are developed by a subcontractor, which are then accepted by Aselsan as prime-contractor and delivered to Aselsan's customer. The changes requested after delivery by the customer are performed by either Aselsan or the subcontractor, according to the details of the subcontracting agreement.

The two software engineering departments had 136 personnel working at the time of writing of this thesis. The software development processes of the two departments are based on ISO 12207 Software Life Cycle Processes standard (ISO, 1995). There is also a measurement framework that is being introduced that is based on the ISO 15939 Software Measurement Process standard (ISO, 2002). The departments have been certified with the AQAP-150 standard of NATO (1997). There is work underway to obtain AQAP-160 certification (NATO, 2001).

There are three main reasons for Aselsan to engage in subcontracting arrangements:

a) The unavailability of personnel. When a project is very large, or the YMM personnel are already tied up in other development projects, Aselsan seeks subcontracting arrangements that will lead to alleviate the lack of manpower to undertake a project.
b) The unavailability of certain expertise to use in a project that calls for it. In order to satisfy the requirements of a project that requires knowledge in a field that is unfamiliar, subcontracting arrangements that result in both the required product and accumulation of knowledge are made.
c) The terms of a contract may dictate cooperation with other firms as subcontractors.

Apart from these three factors, the mission statement of Aselsan states that it will help foster the defense industry in Turkey through the use of subcontractors for defense projects.

In order to evaluate the subcontracting arrangements that Aselsan has made with its subcontractors, some metrics selected according to the criteria described later in this section are required. Below are a summary of metrics that are to be used in the study:

General Information about a subcontracting project:

- Project Duration
- Is source code to be turned over to Aselsan?
- The phase (if any) that the project was taken over by Aselsan.

Metrics that can be computed during the selection phase of a project:

- Number of subcontractors short-listed
- Number of responses to RFPs
- Estimated size of project (KSLOC, number of software, hardware, user interfaces)
- Rating of the subcontractor prior to contract (by Aselsan)

Metrics that can be computed during the contract management phase of a project on regular intervals:

- Size (SLOC, number of function declarations, function points, number of requirements, document page counts)
- Effort (person hours)
    - By Subcontractor
    - By Aselsan
- Conformance to requirements (number of requirements, number of requirements that were satisfied, number of incorrectly implemented requirements, number of unsatisfied requirements, number of changes to the requirements since the initial version)
- Quality
    - Number of Change Requests
    - Number of Accepted Change Requests
    - Ratio of Number of Document Reviews to Number of Documents (including versions of the documents)
    - Fault Density (number of errors per KSLOC)
    - Cyclomatic Complexity
- Budget (dollars)

Metrics that can be computed after the completion of a project:

- Return on investment (as percentage of investment)
- Conformance to project plan / schedule (as percentage of planned duration)
- Conformance to budget (percentage of initial contract value)
- Conformance to requirements (number of requirements, number of requirements that were satisfied, number of incorrectly implemented

requirements, number of unsatisfied requirements, number of changes to the requirements since the initial version)

- Conformance to contract (percentage of clauses that were satisfied, percentage of clauses partially satisfied)
- Rating of the subcontractor after the contract (by Aselsan)
- Size of project (number of software, hardware, user interfaces, SLOC, number of function declarations, function points, number of requirements, document page counts)
- Quality
    - Number of Change Requests
    - Number of Accepted Change Requests
    - Ratio of Number of Document Reviews to Number of Documents (including versions of the documents)
    - Fault Density (number of errors per KSLOC)
    - Cyclomatic Complexity
- Cost of repair/change
    - For the subcontractor in terms of person hours + dollars
    - For Aselsan in terms of person hours + dollars
    - Change cycle efficiency (time to correct an error/ make a change)
    - Modification Complexity (time spent per correction / change)
    - Change Success Ratio (#errors occurring due to previous correction / changes)

It is the aim of this thesis to look for any relationship between the "success" of a project and the metrics that were collected. What is considered a successful subcontracting project is decided through use of some success indicators. These indicators for project success shall be defined in section 3.3.

The metrics that have been suggested here were brought together under the following rationale: The metrics required to properly gauge the effectiveness of a subcontracting agreement consisted of some external software quality metrics for the software resulting from the subcontracting arrangement (as defined by ISO 9126-2 Technical Report, 2003), and the metrics specific to subcontracting such as duration of contract, subcontractor evaluation rating at the beginning of contract, subcontractor evaluation rating at the end of contract and other contract parameters.

For the metrics to be of use, care should be taken to make the data collected objective. For instance, measurements regarding requirements should be based on requirement specification documents of similar detail level in order for measurements from different projects to be comparable. Furthermore, the measurement activities should adhere to a well understood procedure based on standards, such as the ISO 15939 (2002) in order to make the measurement activities uniform and ensure the objectivity of data collected. The quality of the metric data will depend on the quality of the measurement process that is used to collect the data.

The “conformance to requirements” metrics along with “number of errors” and “number of changes” proposed for this case study, cover the requirement and quality aspects of IS system effectiveness presented by Drury and Farhoomand (1998) and other IS success models presented in section 2.5.

The metrics regarding “conformance to requirements” follow from the “suitability metrics” defined by the ISO 9126-2 external software quality metrics report. They aim to determine the stability and the correct implementation of functional requirements for a piece of software.

The metrics regarding “cost of change” are extracted from the “changeability metrics”, again defined by the ISO 9126-2 external software quality metrics report. These metrics aim to determine how easy it is to modify a piece of software once an error has been discovered or some requirement change dictates that a part of the software should be changed.

Some other external quality attributes defined by ISO 9126 such as “usability metrics” which include “learnability, understandability and operability”, are not mentioned directly in the proposed metrics. These metrics would have to involve direct evaluation of the users of the subcontracted software. These attributes are tried to be captured through inclusion of “number of change requests”, for both improvements of existing features and new feature requests.

As was mentioned in Chapter 2, according to ISO-9126, the internal quality attributes defined by ISO-9126 can be used to determine the product quality at the intermediate stages of the development of a software product. They can also be seen as indicators for the quality attributes of the resulting piece of software. In the Aselsan case study, metrics regarding project size and unit test results of functions, functional implementation completeness, functional compliance with requirements, could be collected by the subcontractor at pre-determined intervals to assess the status of the project. These metrics are derived from the "suitability" and "functional compliance" metrics of ISO 9126-3.

Of the internal quality attributes defined by ISO 9126-3, "maintainability" metrics are also considered for the purposes of this case study, as most of the subcontracted software projects are taken over by Aselsan at some point or other during their life cycle. These metrics include measurements about usage of a log, presence of diagnostic functions, record of changes in the source code, number of detected adverse impacts after modifications and presence of test functions.

The ISO 9126-4, "quality in use" metrics have to be evaluated in the actual environment of the software with the real users of the system. This occurs after the installation of the system at the site of operations. Although the metrics that could be collected through tests that would be performed "on site", would probably be beneficial, the collection of these metrics is not practical for the purposes of this study due to the fact that the systems are installed at armed forces facilities

dispersed throughout the country. Therefore, the quality in use metrics are not included in this case study.

The metric of subcontractor evaluation was based on Torchiano's use of maturity in marketplace and market share metrics for selection of COTS software (Torchiano et al., 2002). The metric for the type of subcontracting arrangement was also derived from Torchiano's licensing type metric for selection of COTS software.

As was mentioned in Chapter 2, there are three phases of software subcontracting. These phases are:

1. Looking for subcontractors (selection),
2. Contract Management,
3. Completion of Contract (maintenance).

For these three phases, it would make sense that there are different and overlapping metrics regarding each phase. Therefore, the metrics that have been proposed in this thesis were grouped accordingly.

During the contract management phase, the quality of the system is tried to be measured through use some of the software reliability indicators by Li and Smidts (2003). These are: Requirement Specification Change Requests, Fault Density (number of errors per KSLOC), Cyclomatic Complexity and Code Defect Density.

The metrics collected regarding the subcontracting projects cover only the periods of time during which the project was developed with the subcontractor. If a project was taken over by Aselsan at any point during the project lifecycle, it moves outside the scope of this study. If a project is taken over by Aselsan, this is indicated along with the phase of project in which the takeover occurred by a metric in the table given.

3.3 Software Project Success Indicators

Project success indicators are tools to assess whether a project is considered to be a success. The suggested success indicators for the subcontracting projects in YMM are:

- the ratio of actual development time to estimated development time
- the ratio of actual cost to estimated cost
- rework requests per KSLOC
- Boolean variable whether the project has been approved by the customer

These project success indicators were obtained through interviews with the Software Project Team Leaders in YMM.

Similar project success indicators are also found in the literature. The “Extreme Chaos” article, the successor to the well-known “Chaos Report” by The Standish Group (2001), showed that 23% of IT projects are cancelled while 49% run over

budget. The study divided projects into three categories: Successful, Challenged and Failed. Projects that were "completed on-time and on-budget, with all features and functions as initially specified" were considered successful. Projects that were "completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified" were considered as challenged while cancelled projects were classified as failed. The Standish Group, points to "lack of skilled project management and executive support" as the reason for the failure of most projects. Similarly, the Gartner Institute survey of 2000 shows, that 40% of IT projects fail (Journal of Accountancy, February 2001, p. 24). The report found the primary cause to be ineffectual management and recommended additional training for project leaders.

The "Extreme Chaos" report also states that most of the "successful" projects were over-estimated, meaning more than the required resources have been allocated to them. This suggests better use of metrics should be made for estimating the sizes of projects and allocating resources appropriately.

Karadağ (2003) and Aykol (2003), claim that the failure rates of IT projects revealed by the research conducted by the Standish Group and the Gartner Institute in IT project success is due to the inadequate project management. Karadağ states that despite this fact, very few firms in Turkey make use of the project management methodologies that have been used in the world since 1960s. He states that Türkiye Bilişim Derneği (a major non governmental organization in the Turkish IT sector) has made the promotion of project management methodologies for IT projects a top

priority to decrease the failure rates. Aykol claims that the use of risk management strategies is vital for a project's success.

The use of metrics to track the performance of subcontracting projects helps the project management to better control a project and reduces risks. Therefore the use of metrics can be considered as a measure to ensure project success.

Erener (2003), indicates that there are numerous factors determining whether a software project is successful or not. The author states that success is a function of timetable, budget and software quality. While meeting the project milestones and staying within budget seem obvious success indicators, the quality aspect of software is more complex. According to the author, software quality has two dimensions; one is conformance to requirements, while the other is the ratio of number of defects to the size of code. He also states that not all defects or errors are of equal severity, and gives the error levels of "Show stopper", "Critical", "Important, and "Cosmetic" as a classification schema for errors that a software product exhibits. He states that acceptance criteria for software projects must include defect/size ratios for these different classes of errors.

UKSMA (2000) also uses 4 levels for defect classification, but attaches different labels to the categories. Levels "Critical", "Major", "Minor" and "Cosmetic" are used as different classes to indicate the extent to which a defect affects the success of a software system. It also provides examples from other sources in industry

which yield similar classification schemes. There are numerous other sources in the literature that classify defects found in software into different severity classes.

The results from the interviews among YMM personnel are mostly consistent with these findings from literature as they all stress project completion on schedule and within budget. The approach of the Chaos Report stresses the presence of all initially specified functionality as a success indicator, while interviews at Aselsan stressed the ratio of rework requests with project source code size as an indicator, which was also stated by Erener (2003).

The classification of defects into different severity levels is important, as a software product with five “cosmetic” defects, should not be evaluated as of inferior quality when compared to a product with one “show stopper” error. In the suggested metrics for YMM projects, the errors are classified as major and minor errors. While this gives a better idea compared to having no defect classification, a classification scheme with finer granularity should be applied if it better suits organizational goals. As will be seen in section 3.4.3, for most of the current projects, even the major/minor difference of defects is not available.

While using the ratio of actual development time to estimated development time and the ratio of actual cost to estimated cost success indicators, it is important to keep the scope of the project under study in perspective. If the scope of a project were to be broadened to include some features that were not initially in the requirements, changes to the development time and cost actualizations should be

expected, and the success indicators should be evaluated accordingly. It was observed that none of the projects studied within the scope of this work have undergone such changes of scope.

3.4 Investigation of YMM Subcontracting Projects

3.4.1 Research Method

Interviews were conducted with several employees of Aselsan working in various departments related with the software subcontracting projects to gain insight about the nature of the subcontracting arrangements. Among the interviewed personnel were the members of the software project teams and their Software Project Team Leaders that were responsible for the software that was delivered to the customer as a result of the development efforts by the subcontractor, and quality assurance personnel that were present for the review of the documents and the acceptance tests. The list of interviews conducted is given in Table 1.

Table 1 – List of Interviews

Interviewed Aselsan Personnel
Project 1 Software Project Team Leader
Project 1 Software Developer (Author)
Project 2 Quality Assurance Team Member
Project 2 Software Project Team Leader
Project 2 Software Developer
Project 2 Subcontractor Project Team Leader
Project 3 Software Project Team Leader
Project 3 Software Developer

Apart from the interviews, MST Documentation Center was used for accessing project documents such as software requirement specification documents and software version specification documents for obtaining the values for the metrics. Additionally, with the help of personnel working for the project, other documents from the project directories on the MST intranet regarding the state of projects were examined.

3.4.2 Findings

Project 1 involved a single software development project which performed database access, geographic information access and mapping functionalities. The contract for Project 1 was signed in 2002. Although Project 1's duration was stated as two years in Table 2, the actual development of version 1.0 took about six months. The remainder of the first year saw three minor releases (resulting in version 1.03), containing mostly bug fixes and small usability enhancements. At this point, Aselsan asked for some additional enhancements that the subcontractor deemed outside the scope of the existing contract. The subcontractor's price for the additional enhancements in question was found to be too costly by Aselsan, at which point it was decided to take over the maintenance of the project. Therefore, although there was still another year to go for the maintenance period of the original contract between the subcontractor and Aselsan to expire, Aselsan decided to make the modifications itself since the source code for most of the project was turned over as per the conditions on the contract. The author of this thesis was involved in the meetings with the subcontractor and the development efforts after the project was taken over by Aselsan.

Not all of the requested changes could be performed, as some of the proposed changes required modifications to the libraries that the subcontractor had provided without the source code, in accordance with the contract. Further modifications were made and enhancements were added after the initial wave of enhancements. Some enhancements were made by Aselsan even after the guarantee period of the contract had expired. The most current version of the software is 1.17. The size of the code has increased about 40%. Some further changes are also being considered.

The Software Project Team Leader from Aselsan, who was extensively involved with the subcontracting arrangements for Project 1, classifies the project as a success. She states that the project was delivered to the customer with all of the initial requirements satisfied within six months of signing of the contract. She also states that had there been more time available, it would probably be possible to better analyze the project requirements and specify the features that caused the conflict between the subcontractor and Aselsan in the original contract, thus resulting in a more satisfactory subcontracting arrangement.

Project 2, is comprised of 11 smaller software projects, all being undertaken by the same subcontractor. The project involved creation of a training simulator that simulated the parts of a military system that Aselsan built. Unfortunately, since all the different pieces of software function as a whole, separate metrics were not collected about each piece of software. The measures given in Table 2 are the sum for all 11 pieces of software that were developed by the subcontractor. It is very

difficult to break down the total number into figures for different pieces of software, as data was not collected using this information as a classification criterion.

Another peculiarity with Project 2 is that the number of defects is very large compared to other projects. This is almost certainly due to the fact that the number includes not only the error reports and change requests submitted by Aselsan as was the case for the other projects in the study, but also the number of error reports and change requests resulting from the subcontractor for internal use. Again, there is no way to separate the number of defects / change requests originating from Aselsan or the customer due to the way the data was collected and stored.

An interview with the quality assurance team regarding the ratings for the subcontractors showed that, although an evaluation scheme for potential and existing subcontractors was being used, it was not up to date. This "database" was updated "as necessary"; usually, but not always, when a new subcontracting project was being started. Therefore, for Project 2, the rating for the subcontractor is available at the onset of the project, while the rating at the end of the project is not.

An interview with the Software Project Team Leader of the subcontractor of Project 2 was conducted in order to obtain more concrete figures regarding the individual development projects so that they could be used in comparison with data from the other projects. The interview, however, yielded no further data about individual projects, as the subcontractor did not keep data about defect reports for each software development project. The interview also confirmed that no distinction

could be made between the defects or change requests submitted by Aselsan, and the defects or change requests submitted by the subcontractor employees for internal use.

Project 3 was also made of 13 software development projects, including some developed by Aselsan. For this case study, only three of the subcontracted projects were taken into consideration. The subprojects were chosen due to their having a more thoroughly compiled set of metrics among the other subprojects in Project 3. These are named as Project 3.1, 3.2 and 3.3 respectively.

Project 3.1 had no user interface, and worked “behind the scenes” performing some key communication tasks. Project 3.2 performed some training simulation tasks for the other parts of Project 3. Project 3.3 performed some backup and restore operations as well as providing some conversion functionality for data received on its interfaces.

For Project 3, the subcontractors were determined by the customer at the onset of the contract with Aselsan designated as the prime contractor. Therefore, there was no subcontractor selection process. The project was described as being difficult to manage, as the division of labor among subcontractors was not satisfactory. For instance, the geographical information systems were implemented by more than one subcontractor, each using different technologies indicating a lack of coordination among the subcontractors. Due to these factors and general dissatisfaction with most of the subcontractors and the software that they delivered, the project

management decided not to subcontract any components of the project for the second phase of the Project 3.

The Software Project Team Leader for Project 3 stated that although all of the subprojects appeared to have been delivered on-time and on-budget, there were serious deficiencies with some software which had to be resolved by Aselsan personnel before the customer accepted the software formally.

Project 3.1 was described as a well developed project with minimal number of problems. The Software Project Team Leader for Project 3 states that while there were a small number of errors discovered for this project, this may be due to the fact that the developers of the software checked the software themselves using the log files resulting from the use of the software, and fixed some of the bugs before they became a problem for the rest of the system. The engineers working the project at Aselsan had no complaints about this piece of software, but due to the general decision of not subcontracting any of the components of the project for the second phase, Project 3.1 was taken over by Aselsan.

The engineers working on the Aselsan side for Project 3.2 stated that the software performed a lot of functions that didn't appear in the requirements. This and the fact that the software had a user interface that included a lot of errors are responsible for the high number of change requests for this software.

It was also noted during the interviews that, some of the subcontracting firms involved required that all change requests and bug reports be submitted formally, while others didn't bother with any formalities of documenting the change requests. Therefore, the number of change requests and errors should be evaluated bearing this fact in mind.

In the subsection below, the relationship between project metrics and success indicators is presented in a tabular form.

3.4.3 Suggested Metrics Applied to YMM Subcontracting Projects

The table below provides a comparison of the suggested metrics and the software project indicators described in sections 3.2 and 3.3 for the subcontracting projects of YMM. Other subcontracting projects of YMM had no metrics collected that could be used in this study.

The fields that are marked with DNA denote that data is not available for that metric.

Table 2 - Metrics Applied to Selected Aselsan Subcontracting Projects

Phase	Metric Class	Metric	Project 1	Project 2*	Project 3.1	Project 3.2	Project 3.3
Project Information		Project Duration	2 years	3 Years	3 years	3 years	3 years
		Is source code to be turned over to Aselsan?	Yes	Yes	Yes	Yes	Yes
		The phase (if any) that the project was taken over by Aselsan.	Maintenance	None	Phase 2	Phase 2	Phase 2
Success Indicators		Actual Development Time / Estimated Development Time	1	1.64	1	1	1
		Actual Cost / Estimated Cost	1	1	1	1	1

Table 2 - Metrics Applied to Selected Aselsan Subcontracting Projects – Continued

Phase	Metric Class	Metric	Project 1	Project 2*	Project 3.1	Project 3.2	Project 3.3
		Rework Requests / KSLOC	17 / 120 = 0.14	1277 / 250 = 5.11 **	9 / 101 = 0.09	125 / 30 = 4.17	23 / 31 = 0.74
		Boolean variable whether the project has been approved by the customer	Yes	Yes	Yes	Yes	Yes
Metrics that can be computed during the selection phase of a project		Number of subcontractors short-listed	2	5	Not Applicable (****)	Not Applicable (****)	Not Applicable (****)
		Number of responses to RFPs	2	5	Not Applicable	Not Applicable	Not Applicable
		Rating for the subcontractor	DNA	83 / 100 = 0.83	DNA	DNA	DNA

Table 2 - Metrics Applied to Selected Aselsan Subcontracting Projects – Continued

Phase	Metric Class	Metric	Project 1	Project 2*	Project 3.1	Project 3.2	Project 3.3
Metrics that can be computed after the completion of a project		Conformance to project plan / schedule (as percentage of planned duration)	100	164	100	100	100
		Conformance to budget (percentage of initial contract value)	100	DNA	100	100	100
	Conformance to requirements	the number of requirements that were satisfied / number of requirements	151 / 151 = 1	DNA	71 / 73 = 0.97	80 / 89 = 0.90	147 / 165 = 0.89
		number of incorrectly implemented requirements / number of requirements	0 / 151 = 0	DNA	0 / 73 = 0	5 / 89 = 0.06	2 / 165 = 0.01
		Number of unsatisfied requirements / number of requirements	0 / 151 = 0	DNA	2 / 73 = 0.03	4 / 89 = 0.04	16 / 165 = 0.10
		number of changes to the requirements since the initial version / number of requirements	68 / 151 = 0.45	DNA	5 / 73 = 0.07	17 / 89 = 0.19	19 / 165 = 0.12
	Conformance to contract	percentage of clauses that were satisfied	100	DNA	100	100	100

Table 2 - Metrics Applied to Selected Aselsan Subcontracting Projects – Continued

Phase	Metric Class	Metric	Project 1	Project 2*	Project 3.1	Project 3.2	Project 3.3
		percentage of clauses partially satisfied	0	DNA	0	0	0
		Rating for the subcontractor	DNA	DNA	DNA	DNA	DNA
	Size of project	SLOC	120K	250K	101K	30K	31K
		number of function declarations	2350	DNA	2612	479	691
		number of software interfaces	2	DNA	2	4	7
		Number of hardware interfaces	0	DNA	0	0	0
		Number of user interfaces (windows)	99	DNA	0	60	0
		Number of requirements	151	990	73	89	165
		Total document page count	1411	DNA	DNA	DNA	DNA
	Number of errors (***)	major errors per KSLOC	16 / 120	260 / 250	9 / 101	125 / 31	20 / 30
		Minor errors per KSLOC	DNA	508 / 250	DNA	DNA	DNA
	Quality	Number of change requests	17	446	9	125	23

Table 2 - Metrics Applied to Selected Aselsan Subcontracting Projects – Continued

Phase	Metric Class	Metric	Project 1	Project 2*	Project 3.1	Project 3.2	Project 3.3
		Number of Document Reviews / Number of Documents (including all versions of the documents)	2 / 29 = 0.07	DNA	14 / 20 = 0.70	15 / 19 = 0.79	17 / 23 = 0.74

* 11 individual software products

** Figure includes rework requests for all subprojects and internal rework requests of the subcontractor.

*** Major/Minor Error Data was unavailable for some of the projects.

**** Subcontractors dictated by the customer

CHAPTER 4

DISCUSSION OF FINDINGS

This study examines three subcontracting cases involving fifteen software development projects that have been carried out in YMM during the past few years. The outcomes and success indicators of the three cases differ as Project 2 moves into phase 2 with the subcontracting company, Project 1 has been taken over by Aselsan during the maintenance phase due to the cost of development of new features, and Project 3 has been taken over by Aselsan due to dissatisfaction with the performance of the subcontractors.

The aim of this thesis was to investigate, via a case study, software subcontracting success factors. A relationship was sought between the metrics that have been suggested in section 3.2, and the project success indicators given in section 3.3. The metrics described in section 3.2 were classified in four categories: general metrics for the entire project, metrics for the selection phase of the project, metrics for the contract management phase of the project, and lastly metrics that can be computed after the completion of the project. The data collected for this case study does not

include data regarding the contract management phase of projects, as no data was available for this phase.

In addition to the metrics collected, to gain insight about the context of the subcontracting arrangements, interviews with YMM personnel were conducted and the specifics of the subcontracting deals were recorded.

The success indicators described in section 3.3 were based on findings from the literature and views of the Software Project Team Leaders in YMM. These were found to be: the ratio of estimated vs. realized project schedule, the ratio of estimated vs. realized project cost, the number of rework requests per KSLOC, and the Boolean variable of whether the project was approved by the customer.

Of the three subcontracting arrangements, Project 1 concerned a single software development project, Project 2 had 11 software development subprojects, and Project 3 had 13 software development subprojects. Project 2 was undertaken by a single subcontractor and data about the individual projects is not available. Therefore a comparison of metrics of Project 2 subprojects and other subprojects was not possible. Project 3 was built by several subcontractors, including some parts built by Aselsan. Only three of the subcontracted subprojects were chosen due to the completeness of their data.

After the examination of Projects 1, 2 and 3 in this case study, the following relations among the metrics and success indicators have been suggested:

Of the four success indicators, only “change requests per thousand lines of code” showed difference among different projects. Except for Project 2, the ratios actual development time to estimated development time, actual cost to estimated cost and the approval of the project by the customer were identical for all projects. Therefore, the only meaningful comparison among the success indicators for the projects is among the figures for the rework requests.

Due to the nature of subcontracting processes in Aselsan, the cost of a subcontracting project is usually fixed by the contract. If there are any unexpected costs, they are born by the subcontractor. Therefore, it is to some extent expected that all of the subprojects cost exactly as much as they were estimated to cost. Although the amount paid to the subcontractor for a project doesn’t change, the amount of money and effort expended by Aselsan could be considered as a cost of the subcontracting project. These costs occur due to the fact that although a project is subcontracted, since Aselsan remains the prime contractor, it is still responsible for the success of the project. It may usually be the case that the subcontracted software has interfaces with some other software being developed by Aselsan and the integration of the subcontracted subprojects into the larger project requires time, money and effort by Aselsan. Unfortunately, there is currently no way to measure the estimates and the actual costs of the time, money and effort incurred by Aselsan for subcontracting projects. For large projects, however, these costs probably remain a fraction of the value of the subcontract, and not affect the overall cost very much.

As for the “conformance to schedule” and “acceptance by the customer” success indicators; usually most of the projects are delivered on time and are accepted by the customers. But in order to make the project delivery deadline, some features may be taken out, the written code may be less thoroughly tested, and some of the features may not exactly work as intended. The customer is usually placated through the promise of service releases to remove errors, and addition of the originally intended functionality. Again, since the delivery time and the required features are stated in the contract that Aselsan makes with the subcontractor; the subcontractor has to bear the costs of correcting the errors and adding the missing functionality. But after the project deadline has passed, the time and effort spent by the subcontractor for an “accepted” project is usually not as much as the time and effort spent before the deadline. Therefore, the only viable indicator for project success becomes the ratio of the number of defects to the software size.

Project 2 was turned down by the customer during its initial acceptance tests. This caused a difference between the estimated duration of the project and the actual duration of the project. Interestingly, the second acceptance tests almost a year later, (a delay of 64%) were successful and it is the only subcontracting agreement in this case study that is still continuing. The fact that the subcontract still continues into another phase of project, may indicate that the subcontracting relation was a success.

The metrics that differed from each other are given in the table below along with their relation to the success indicator defect ratio. Figures 1 through 10 represent graphically the values of the metrics against the indicator.

Table 3 - Relationship between Metrics and Success Indicator

Metric:	**Success Indicator: Rework Requests / KSLOC (lower is better)**	**Figure**
Percentage of Requirements that were satisfied	Positive Relation (i.e. Success Increases with higher percentages)	1
Percentage of Requirements that were incorrectly implemented	Negative Relation	2
Percentage of unsatisfied requirements	Inconclusive	3
Ratio of changes to requirements to number of requirements	Inconclusive	4
Program Size (KSLOC, number of function declarations)	Inconclusive	5, 6
Number of software interfaces	Inconclusive	7
Number of User interfaces (windows)	Inconclusive	8
Number of errors / KSLOC	Negative Relation	-
Number of requirements	Inconclusive	9
Number of Document Reviews / Number of Documents	Inconclusive	10

It appears from Table 3 that the metrics regarding the state of requirements for a project have an effect on the project success outcome. (Figures 1, 2, 3) Especially, the percentage of incorrectly implemented requirements shows an almost linear

relation to the defect ratio. This suggests that properly defined requirements should be sought by subcontracting project management. The state of requirements could be a good indicator to watch during the contract management phase to gain understanding about the future success of the project.

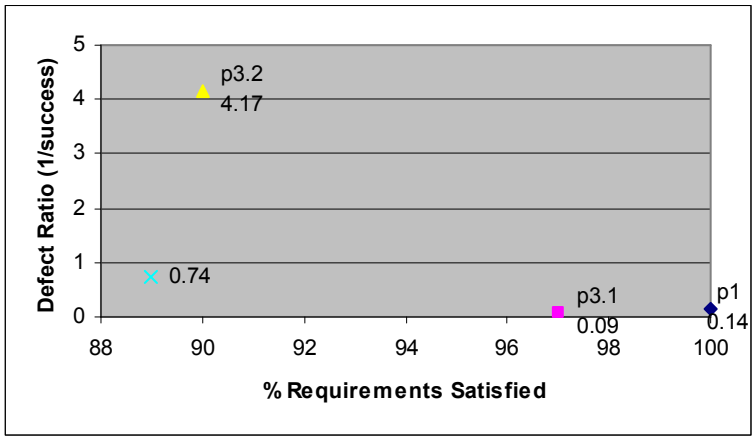

Figure 1 – Percentage of Satisfied Requirements

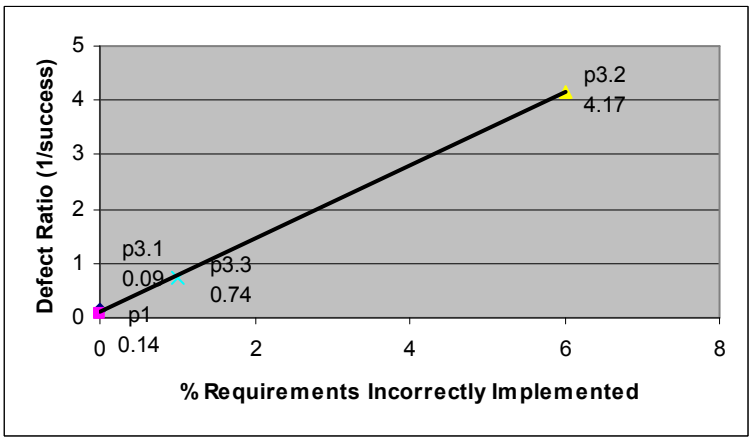

Figure 2 – Percentage of Incorrectly Implemented Requirements

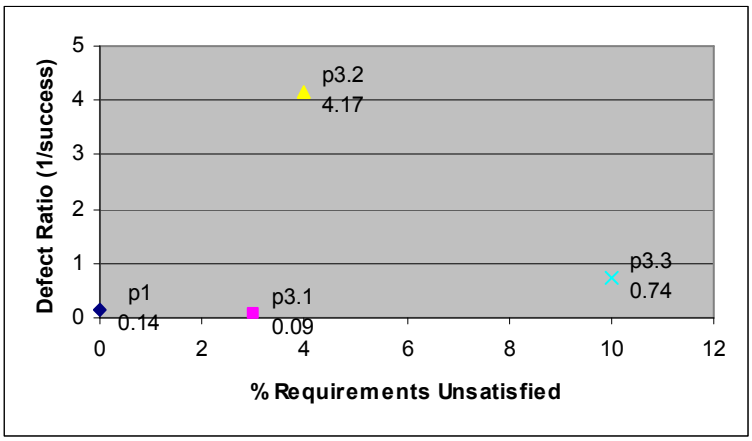

Figure 3 – Percentage of Unsatisfied Requirements

The ratio of changes to requirements to number of requirements does not present a clue about the probable success of a project. Project 1, one of the more "successful" projects, has the highest amount of changes to requirements after its initial set of requirements. (Figure 4)

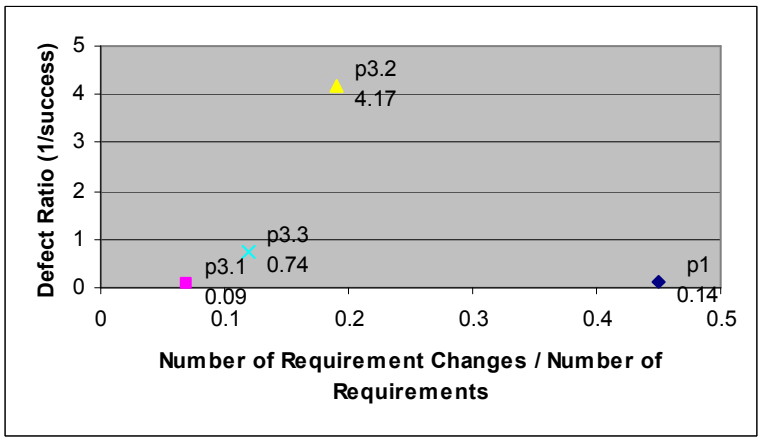

Figure 4 – Number of Requirement Changes / Number of Requirements

The data shows that larger projects (Project 1 with 120K lines of code and Project 3.1 with 101K lines of code), have lower rework requests per thousand lines of code ratios (0.14 and 0.09 respectively), while smaller projects have higher values for this success indicator. (Figure 6) A similar trend is apparent in the number of function declarations that the source code contains. (Figure 5) This may suggest that, although a project is larger, and requires greater effort to accomplish, it doesn't necessarily have to be "less successful" compared to simpler (or smaller) projects. While this may be due to higher importance attached to the contract by the subcontractor, and hence, a higher level of quality achievement, this point definitely deserves further study.

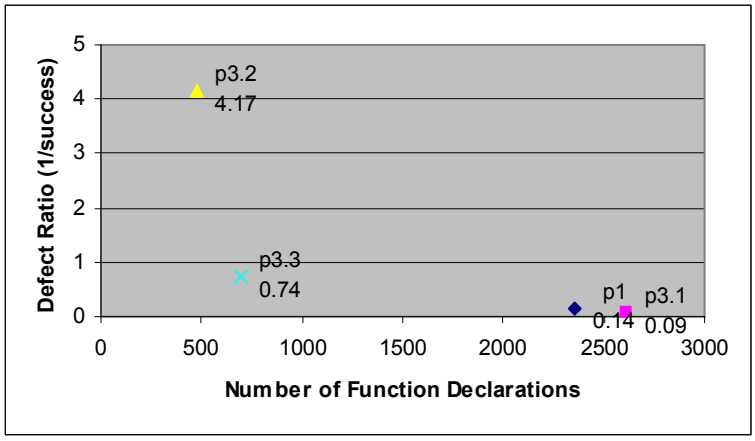

Figure 5 – Number of Function Declarations

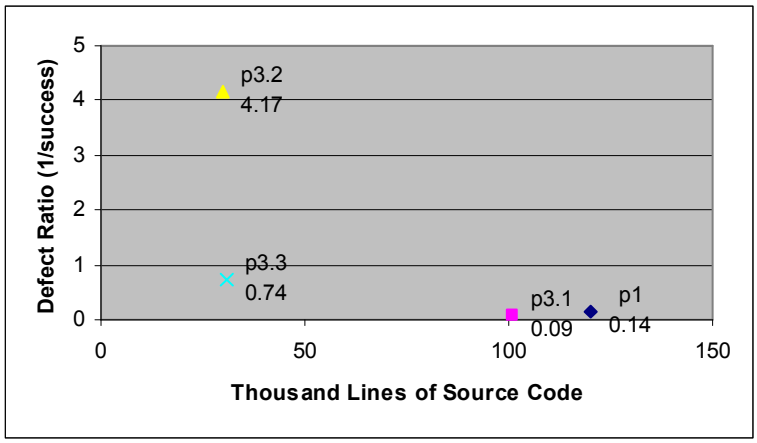

Figure 6 – Thousand Lines of Source Code

While number of software interfaces does not point to a clear relationship to project success, two most "successful" projects also have the lowest number of software interfaces. (Figure 7)

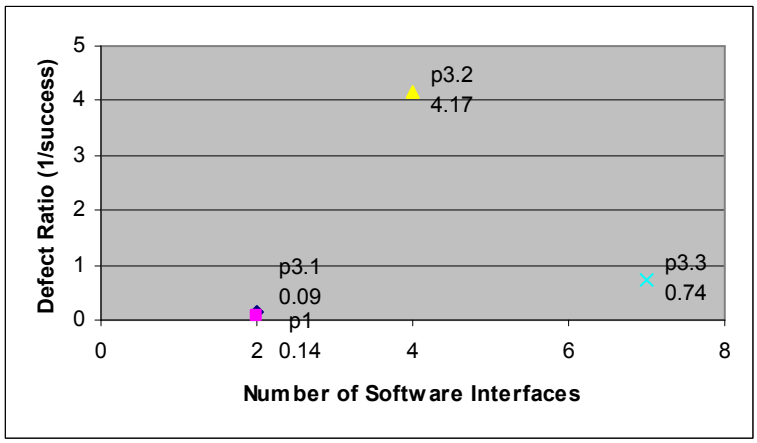

Figure 7 – Number of Software Interfaces

Number of requirements, which can also be seen as a measure of project complexity, fails to give an obvious indication about the success of a project.

Project 3.2, with the highest defect ratio has lower number of requirements compared to more “successful” Projects 1 and 3.3. (Figure 8)

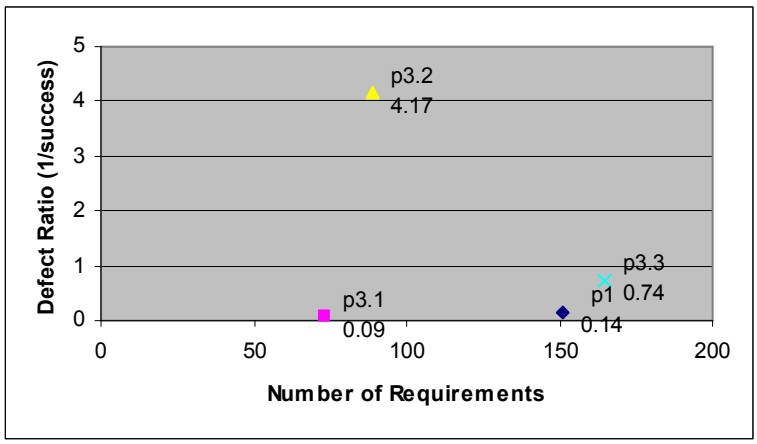

Figure 8 – Number of Requirements

It is notable that of the four software products considered, the presence of a user interface does not directly affect project success. This may appear surprising as the user interface color schemes, placement of user interface elements and the workflow of different windows are usually susceptible to change. Project 1 has 99 windows, while Project 3.2 has 60. However, the number of rework requests per thousand lines of code for Project 3.2 (4.17) is much higher compared to Project 1 (0.14). Project 3.3, which lacks a user interface, has a higher rework requests per thousand lines of code ratio (0.74) than Project 1, and therefore appears “less successful”. (Figure 8)

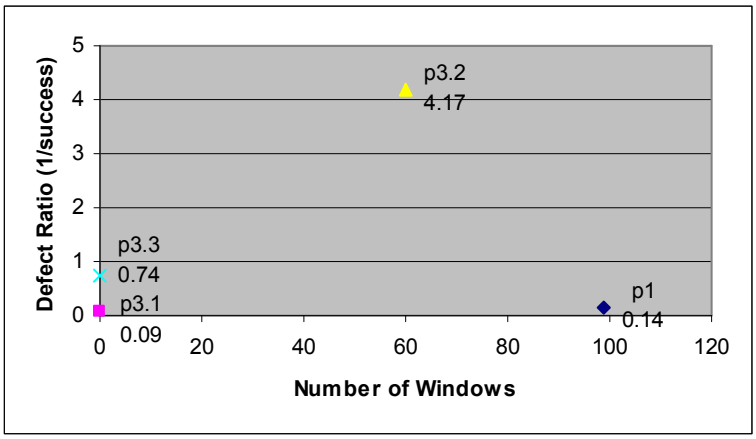

Figure 9 – Number of Windows

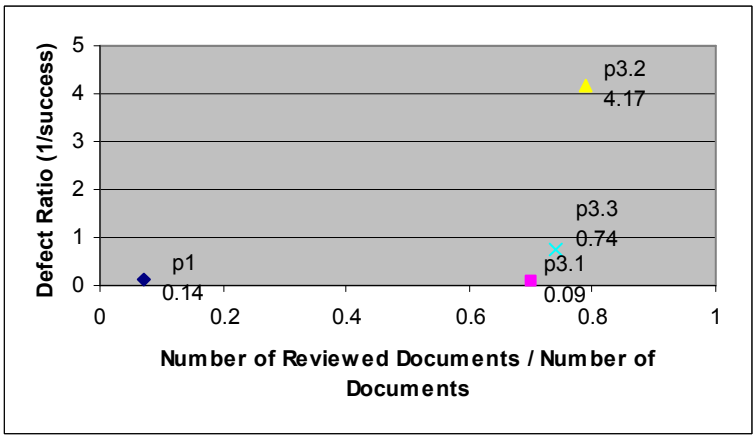

Figure 10 – Number of Reviewed Documents / Number of Documents

The ratio of number of documents that were reviewed to number of documents received from a subcontractor presents no discernible relationship to the project success. (Figure 10)

One row in Table 3 shows that the number of errors per thousand lines of code has a negative relationship with the project success ratio. This can not be considered an

indicator, as for most of the projects it is the same thing as the number of change requests per thousand lines of code success indicator.

Project 3.2 has a relatively high defect ratio compared to the other projects in the case study, and therefore appears less successful. As was described in section 3.4.2, the requirements of this subproject didn't accurately reflect the functionality that the software performed, and that the requirement specification for Project 3.2 was inadequate. Since requirements are a prerequisite for design and implementation of software, it is safe to assume that inadequately specified requirements may lead to a lot of rework requests. Many examples from the literature consider conformance to requirements as an indicator of quality, which was also the reason for conformance to requirements metrics to be collected in this case study. Therefore high value of the rework requests per thousand lines of code success indicator for Project 3.2 may be attributed to the mentioned inadequacy of proper requirement analysis.

A quick look at the Table 2 presented in section 3.4.3 shows that it contains not as many metrics as were suggested in section 3.2. This is due to the fact that some of metrics for the three projects are not available. This is especially noticeable in the estimates for an upcoming project during the subcontractor selection phase, the metrics during the development phase, and the metrics regarding the costs of changes. Although YMM is trying to incorporate the measurement of its business processes into the process of developing software, this still is a relatively new effort. Even when measurement is performed for a project, measurement efforts are usually limited to the projects or project parts that are being developed at YMM.

Therefore the subcontracted projects do not currently participate in the measurement process. The metrics in Table 2 are derived from the rework requests, defect management tools and any existing documents regarding the subcontracted software projects.

CHAPTER 5

CONCLUSION

Before presenting the conclusions derived from this study, it must be noted that this thesis is based on a case study, and does not aim to make any generalizations about the metrics to be used to ensure the success of any given subcontracting arrangement. It only uses the data from three subcontracts involving fifteen software development projects in YMM. The data used in the study covers only a part of the proposed metrics as YMM has not incorporated the collection of such metrics to its subcontract management procedures. The thesis aims to investigate the possibility of better controlling subcontracting arrangements in light of the existing data and experience captured through interviews with YMM personnel.

The unavailability of a complete set of data for the proposed metrics undermines the certainty of the relations that are suggested. Especially the lack of individual project data for Project 2, along with the fact that it was not possible to isolate the number of change requests submitted by Aselsan was not available, limits the credibility of the relations found using four software development projects (Projects 1, 3.1, 3.2, and 3.3).

The unavailability of metrics during the development phases of the projects used in the study is an important limitation. As described in Chapter 1, close relations with the subcontractors should be sought for better management of the contract through collection of metrics during the contract management phase. This will enable the management to gain understanding about the course that a subcontracting agreement is taking and make any required adjustments to the subcontracting processes.

It must also be noted that, while the same metrics for all projects have been compiled, the projects were different in nature, and were developed by different subcontractors. For example, Project 1 had 6 months for requirements elicitation, design, coding, testing, and delivery to the customer. On the other hand, the software developed for Project 3 had development durations of 3 years. There was, however, the challenge of communicating the needs of the different subprojects to different subcontractors to create a system that used all of the software. Therefore it must be noted that the results and success of a subcontracting agreement may depend on the context that it was formed in.

The selection method of the subcontractors was different for the projects as well. Project 3 had its subcontractors dictated to Aselsan by its customer, while for the other projects, Aselsan chose the subcontractors. It may be worth investigating the selection method for the subcontractors as an issue affecting the success of subcontracting agreements.

The data collected in the study to compare the success rates of different projects can be deemed as objective, as the data is based on quantifiable results and YMM procedures ascertain the adherence of products and documents submitted by a subcontractor to a certain level of standard and uniformity.

The relationship observed between the metrics and the success indicators may warrant an investigation of YMM's integration of a metrics based control framework into its subcontracting arrangements. In order to effectively use the metrics suggested in section 3.2, a framework for the measurement of the metrics could be constructed, and incorporated into the management processes of future software subcontracting projects. As indicated by Eralp (2004), measurements should be consistent with organizational goals if they are to succeed. Therefore the list of metrics suggested in this study could be considered as a starting point that investigates whether collection of metrics could be beneficial to the organization regarding software subcontracting activities.

The following is a list of possible hypotheses that may be derived from the discussion in Chapter 4:

1. There seems to be a positive relationship between the requirements and the success of a subcontract. A greater number of correctly implemented requirements indicate a more successful project, while high number of incorrectly implemented requirements indicates a less successful project.

2. Larger projects have less number of change requests per thousand lines of code, and are considered to be more successful. This may be attributable to greater importance attached to larger projects by the subcontractors. In turn, it may be assumed that smaller projects may not always get the time, effort and care that the large projects get and thus become less successful.
3. While size defined by code or number of function declarations favors larger projects, the more successful projects are the ones that have less number of interfaces to other software.
4. It may be healthier for a subcontracting arrangement to reject a product with missing features rather than accepting with assurances of updates during the maintenance period for the missing features, in order to keep the subcontractor fully committed to the project.

Further research may focus on creating a framework for following the subcontracting metrics in order to control the effectiveness of a software subcontracting arrangement. To start such an investigation for integrating the collection of metrics into subcontract management processes, the metrics given in section 3.2 can be taken as a basis. Priority could be given to the metrics presented in Table 3 that appear to point to a relationship with the success of a subcontracting project. For example, tracking the state of requirements could be a good indicator for the future success of a project.

In order to further investigate the relation of metrics to success indicators, a panel of project managers could consider the metrics suggested here to evaluate whether

they are appropriate for the company's organizational goals, adding other metrics or removing the existing metrics. The panel would convene periodically to assess and fine tune the metrics to be collected and over time arrive at a stable framework for effective software subcontract management.

It must be underlined that suggested metrics include metrics for the selection and contract management phases of the subcontracting projects. It would seem wise that for the metrics to be of any use to YMM and other organizations on subcontracting agreements, the metrics collected during the selection and contract management phases should be kept under watch, and if any values appear to indicate a problem, adjustments should be made to the subcontracting process. Therefore, the metrics can be used not just for "post-mortem" analysis of how well the subcontracting arrangement has worked out, after the duration of the contract has expired; but actively during the contract management.

The compilation of the metrics and success indicators for subcontracting projects would, in time, lead to the compilation of a subcontracting database, which could be used for comparing and contrasting the active subcontracting agreements with the successful and unsuccessful examples from the past. The compilation of historical data may invite use of statistical methods for effectiveness analysis of the subcontracting arrangements as well, which is beyond the scope of this study.

The selection phase metrics may indicate to management whether the bids for a proposed contract are adequate, whether the bidders will be up to performing the task and thus lead the decisions for awarding the contract.

The contract management phase metrics may indicate whether the project of a certain size is developing at the rate it should be. If there is a tendency to miss the deadlines, the subcontractor can be contacted to discover what the problem is, whether it can be solved or maybe if another subcontractor should be sought for the project.

The metrics that are collected after the completion of a project will provide a basis for evaluation of the subcontractor for future projects and also become a historical fact that can be used for estimation purposes in projects of similar attributes.

As is stated in section 3.2, the effectiveness of the data used in such a framework depends on how the data is collected. In order for the metrics to be effective, a measurement process should be established and adhered to. Standards such as the ISO 15939 Software Measurement Process (2002) could be used to create a measurement procedure in the organization that will collect data about the metrics.

Further research can look into the relationship that requirement specification has with software subcontracting arrangements. Establishment of criteria for ensuring that the requirements of software were well-understood by the subcontractor could

enable the subcontracting relations to be more successful from both the customer's and the subcontractor's perspective.

Another interesting point to investigate may be the relationship among the size and complexity of software and its success. It seems intuitive that larger and more complex software will also be harder to implement and therefore may require greater effort from the subcontractor. However, smaller projects, while requiring less effort, may suffer from inadequate attention of the subcontractor and become failures. This dynamic between size, complexity and the importance attributed to software may offer insight into subcontracting and software development in general.

As stated in this study, the "success" of an effective subcontract management arrangement depends on contractual success (such as completing the project on time and within budget) as well as the software quality of the resulting software product. While adherence to stated contract parameters plays an important role in business transactions; the success of the information system that the subcontracted software product becomes a part of is dependent on the quality of the software product itself. Therefore, the quality of the resulting software product has an effect on the IS success.

The usefulness of information systems is dependent on the process through which the information system is built. Therefore defining the requirements of an information system properly and correctly, communicating the parameters of the

envisioned system to those who build the system, and validating the resulting system are essential to IS success.

When software subcontracting is considered, it is noticeable that it involves both communication of the envisioned system to those who build the system and validation of the resulting software product. Therefore, software subcontracting plays an important part in the process of creating a successful information system, and the process can be more manageable and predictable through use of a metrics based control framework.

REFERENCES

Assmann, D. & Punter, T. (2004). Towards partnership in software subcontracting. Computers In Industry 54 (2), 137-150.

Aykol, M. M. (2003). BT Projeleri Neden Başarısız Olur?. TBD Dergi, November 2003. Available: http://dergi.tbd.org.tr/yazarlar/17112003/meric_aykol.htm

Benbasat, I., Goldstein, D. K., Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. MIS Quarterly. 369-386.

Bertoa, M.F. & Vallecillo, A. (2002). Quality Attributes for COTS Components. Proceedings of the 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002). Malaga, Spain, June 2002.

DeLone, W. H. & McLean, E. R. (1992). Information systems success: The quest for the dependent variable. Information Systems Research, 3(1). 60-95.

DeLone, W. H., McLean, E. R. (2003). The DeLone and McLean Model of Information systems success: A Ten-Year Update. Journal of Information Systems, Spring, Vol.19. 9-30.

Drury, D.H., Farhoomand, A.F. (1998). A Hierarchical Structural Model Of Information Systems Success. INFOR, Vol.36, No.1/2, February-May 1998, 25-40.

Eralp, Ö. (2004). Design and Implementation of a Software Development Process Measurement System. (unpublished). Master Thesis submitted to the Graduate School of Natural and Applied Sciences of Middle East Technical University, supervised by Prof. Dr. Semih Bilgen.

Erener, Ö. (2003). Yazılım Projelerinde Başarı. BTinsan, September 2003. Available: http://www.btinsan.com/1114-03.asp (Turkish)

Hyatt, L. E. & Rosenberg L. H. (1996). A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality. Software Product Assurance Workshop, 19–21 March, ESTEC, Noordwijk, The Netherlands, 1996.

ISO. (1995). Information technology — Software life cycle processes ISO/IEC 12207:1995. Geneve, Switzerland: International Standards Organization.

ISO. (1998). Information technology — Software process assessment ISO/IEC TR 15504:1998. Geneve, Switzerland: International Standards Organization.

ISO. (2002). Information technology — Software Engineering — Software Measurement Process ISO/IEC 15939:2002. Geneve, Switzerland: International Standards Organization.

ISO. (2003). Software engineering — Product quality — Part 2: External metrics ISO/IEC TR 9126-2:2003(E). Geneve, Switzerland: International Standards Organization.

ISO. (2003). Software engineering — Product quality — Part 3: Internal metrics ISO/IEC TR 9126-3:2003(E). Geneve, Switzerland: International Standards Organization.

ISO. (2003). Software engineering — Product quality — Part 4: Quality in use metrics ISO/IEC TR 9126-4:2003(E). Geneve, Switzerland: International Standards Organization.

Journal of Accountancy (2001). Research shows high failure rate on IT projects. Journal of Accountancy, February 2001. 24.

Kaplan, B. & Dennis, D. (1988). Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study. MIS Quarterly. 571-583.

Karadağ, L. (2003). Bilişim Toplumunda Proje Yönetiminin Stratejik Önemi ve Bilişim Projeleri Yönetimi Çalışma Grubu. TBD Dergi, November 2003. Available: http://dergi.tbd.org.tr/yazarlar/03112003/levent_karadag.htm.

Li, M. Smidts, C.S. (2003). Ranking Of Software Engineering Measures Based On Expert Opinion. IEEE Transactions on Software Engineering, vol. 29, no. 9, September 2003.

Myers, B. L., Kappelman, L.A., Prybutok, V. R. (1997). A Comprehensive Model for Assessing the Quality and Productivity of the Information Systems Function: Toward a Contingency Theory for Information Systems Assessment. Information Resources Management Journal, Winter, 1997.

NATO. (1997). NATO Quality Assurance Requirements for Software Development, AQAP-150, Edition 2. North Atlantic Treaty Organization.

NATO. (2001). NATO Integrated Quality Requirements for Software throughout the Life Cycle, AQAP-160, Edition 1. North Atlantic Treaty Organization.

Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C. (1993). Capability Maturity Model for Software, Version 1.1. Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403, February 1993.

Perry, D.E., Sim, S.E., Easterbrook, S. M. (2004). Case Studies for Software Engineers. Proceedings of the 26th International Conference on Software Engineering (ICSE'04), 2004.

Seddon, P.B. (1997). A Respecification and Extension of the DeLone and McLean Model of IS Success. Information Systems Research (8:3), September 1997. 240-253.

Software Engineering Institute. (2002). Software Acquisition Capability Maturity Model, Version 1.03. CMU/SEI-2002-TR-010, March 2002.

Software Engineering Process Office. (2000). Contractor Acquisition and Performance Monitoring Process For Software Contracts. Software Engineering Process Office, D12 Space and Naval Warfare Systems Center.

The Standish Group. (2001). Extreme Chaos. Available: http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf

Torchiano, M., Jaccheri, L., Sørensen, C. F., Wang, A. I. (2002). COTS Products Characterization. 14th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'02), Ischia, Italy, July 15-19.

United Kingdom Software Metrics Association (UKSMA) (October 2000) Quality Standards Defect Measurement Manual, Release 1.a.