1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**REASONS FOR SCHEDULE OVERRUNS IN
SOFTWARE DEVELOPMENT PROJECTS**

by

SUBRAMANI GOPALAN

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements of the degree of**

**Master of Management Studies
School of Business**

**Carleton University
Ottawa, Ontario**

July 13, 1995

© copyright

1995, Subramani Gopalan

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-08993-2

Canada

Name SUBRAMANI GOPALAN

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

MANAGEMENT

SUBJECT TERM

0452

U·M·I

SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Literary Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523

Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578

Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Botanics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996

Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Palaeobotany	0345
Palaeontology	0426
Paleontology	0418
Paleozoology	0985
Polymology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575

Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences	
Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463
Applied Sciences	
Applied Mechanics	0346
Computer Science	0984

Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0431

**The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis**

**REASONS FOR SCHEDULE OVERRUNS IN
SOFTWARE DEVELOPMENT PROJECTS**

submitted by Subramani Gopalan, B.Sc., M.B.A.

**in partial fulfillment of the requirements for
the degree of Master of Management Studies**

Thesis Supervisor

Director, School of Business

Carleton University

September 5, 1995

To my Amma and Appa

ABSTRACT

The objective of this research was to examine the relationship between the importance of the reasons for schedule overruns and software complexity. Schedule overrun was measured as the number of weeks by which the actual duration of the development project exceeded the estimated duration. Number of source lines of code, the number of modules developed for the software product and the average number of source lines of code per module were used as proxies for software complexity.

Eleven reasons for schedule overruns were identified from the literature review and from interviews with experts in software development. A questionnaire was used to gather data from project leaders and senior software engineers of 41 software development projects. The data gathered included the importance of the reasons for schedule overruns, measures of the three proxies for software complexity, magnitude of schedule overruns and the consequences of schedule overruns.

The most important reasons for schedule overruns were found to be: 1. incorrect estimation of development time, 2. changes in specifications and 3. inability to detect problems early. The importance of these three reasons was not correlated to any of the three software complexity measures.

The magnitude of schedule overruns expressed as a percentage of actual and estimated durations increased as the average number of SLOC per module increased. There was, however, no correlation between the absolute magnitude of schedule overruns and the three measures of complexity.

The magnitude of schedule overruns was related positively with problems with organizing project teams and negatively with lack of experience of the team.

The magnitude of schedule overruns was not correlated with the size of the project in relation to projects normally undertaken by the company.

Schedule overruns were found to affect project cost the most, followed by customer satisfaction, financial success and the progress of dependent projects.

Recommendations to managers on how to reduce schedule overruns as well as suggestions for further research are made.

ACKNOWLEDGMENTS

I thank the School of Business for providing me the opportunity to do this research.

I am deeply indebted to Prof. Antonio J. Bailetti, my supervisor, for his continued interest, financial support, invaluable comments and guidance throughout the writing of this thesis and for providing an environment rich in learning opportunities.

I thank Professors John Callahan, Moses Kiggundu, Shirley Mills, and Oscar Hauptman for their comments.

Dr. Ikuko Webster's invaluable assistance with the statistical analysis of the data is much appreciated.

The time and effort of all the software engineers and software project managers who provided the data and suggestions are gratefully acknowledged.

I am grateful to Professors Francis Cherunilam, N. Ranganathan and P.N. Poduval of Cochin University of Science and Technology, India, for encouraging me.

Ms. Jean Blair's commitment and effort in helping me is much appreciated.

I thank my wife Shuba and my dearest Shyamie for their understanding and affection.

I thank Ms. Cindy Dunn whose kindness and commitment to Shyamie enabled me to finish my thesis without undue schedule overruns.

TABLE OF CONTENTS

ACCEPTANCE SHEET	ii
ABSTRACT	iv
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES	xii
LIST OF APPENDICES.....	xiii
CHAPTER 1 INTRODUCTION	1
1.1 RESEARCH QUESTIONS	1
1.2 ORGANIZATION OF RESEARCH QUESTIONS.....	4
1.3 COMPARISON OF THIS RESEARCH WITH EXISTING RESEARCH STREAMS	6
1.4 CONTRIBUTION.....	8
1.5 ORGANIZATION OF THE THESIS	9
CHAPTER 2 LITERATURE REVIEW	11
2.1 REASONS FOR SCHEDULE OVERRUNS	11
2.2 SOFTWARE COMPLEXITY.....	20
2.2.1 Definitions of software complexity	20
2.2.2 Measures of software complexity	21
2.2.3 Software complexity measurement.....	24
2.2.4 Software complexity and development duration.....	25
2.3 SUMMARY OF SELECTED ARTICLES	26
2.3.1 Studies on product development projects	26
2.3.2 Studies on software development projects.....	33
2.3.3 Software complexity	41
CHAPTER 3 RESEARCH DESIGN	55
3.1 RESEARCH QUESTIONS	55
3.2 SAMPLE SELECTION	56
3.2.1 Population.....	56
3.2.2 Sampling method	57
3.2.3 Sample size	58
3.3 UNIT OF ANALYSIS	58
3.4 RESPONDENTS.....	58
3.5 DATA COLLECTED	59
3.5.1 Project data.....	59
3.5.2 Software complexity data	60

3.5.3 Schedule overrun data	62
3.5.4 Reasons for schedule overruns included in the questionnaire	63
3.5.5 Request for a copy of the findings	68
3.6 DATA COLLECTION METHOD	68
CHAPTER 4 DATA ANALYSIS	71
CHAPTER 5 RESULTS	79
CHAPTER 6 DISCUSSION OF RESULTS	113
6.1 SUMMARY OF IMPORTANT FINDINGS	113
6.2. SCHEDULE ESTIMATION PROCEDURES	119
6.3 MANAGEMENT OF THE CHANGES IN SPECIFICATIONS	122
6.4 FRONT-END DEVELOPMENT PROCESS	125
6.5 ADDRESSING COMPLEXITY	128
6.6 ORGANIZATIONAL LEARNING	129
6.7 RESOURCE ALLOCATION	131
6.8 PERFORMANCE OF SOFTWARE DEVELOPMENT TEAMS	132
CHAPTER 7	133
CONCLUSION	133
7.1 RECOMMENDATIONS TO MANAGERS	133
7.1.1 Maintain schedule management database	134
7.1.2 Build prototypes before full scale development starts	135
7.1.3 Increase the development team's knowledge of the problem domain	135
7.1.4 Inform stake holders early of changes in specifications	136
7.1.5 Detect problems early	137
7.1.6 Implement a corporate level strategy to deal with increases in software complexity	137
7.1.7 Verify assumptions	138
7.1.8 Build high performance software development teams	139
7.2 SUGGESTIONS FOR FURTHER RESEARCH	139
7.3 LIMITATIONS OF THE STUDY	141
BIBLIOGRAPHY	143

LIST OF TABLES

Table	Description	Page
1	Comparison Between the Research in This Thesis and Past Research Streams in Product Development	7
2	Reasons for Schedule Overruns Identified in the Literature	18
3	Reasons for Product Development Delays - Gupta and Wilemon (1990)	28
4	Reasons for Schedule Slips and Budget Overruns - Thamhain and Wilemon (1986)	32
5	Classification of Reasons for Effort and Schedule Overruns - van Genuchten (1991)	39
6	Intercorrelations Among Complexity Measures - Curtis Sheppard and Milliman (1979)	49
7	Correlation Between Performance and Complexity Measures - Curtis, Sheppard and Milliman (1979)	51
8	Relationship Between the Reasons for Schedule Overruns Found in the Literature and the Reasons Included in the Questionnaire	64
9	Frequency Distribution Showing the Number of Responses Received from Each Software Development Organization	80
10	Summary of the Characteristics of the Data in the Sample	85
11	Respondent Rating of the Reasons for Schedule Overruns	88
12	Spearman Rank Order Correlation Between the Importance of Each Reason for Schedule Overruns and Complexity	92

LIST OF TABLES (Contd...)

Table	Description	Page
13	Pearson Correlation Between the Magnitude of Schedule Overruns, Magnitude of Schedule Overruns Expressed as a Percentage of Estimated and Actual Durations and Complexity	94
14	Spearman Rank Order Correlations Between the Importance of a Reason for Schedule Overruns and the Magnitude of Schedule Overruns	96
15	Spearman Rank Order Correlation Between the Magnitude of Schedule Overruns and Size of Project Relative to the Size of Projects Normally Undertaken by the Company	98
16	Respondent Rating of the Consequences of Schedule Overruns	100
17	Summary of Responses to Question 3.4	118

LIST OF FIGURES

Figure	Description	Page
1	Organization of Research Questions	5
2	Software Maintenance Project Costs Conceptual Model	45
3	Main Lines of Business of the Organizations that had Projects Included in the Sample	81
4	Number of Employees in the Software Development Organizations that had Projects Included in the Sample	82
5	Frequency Distribution of the Size of Projects Included in the Sample Relative to the Size of Projects Normally Undertaken by the Company	84
6	Estimated and Actual Durations and Magnitude of Schedule Overruns of the Projects in the Sample	87
7	Respondent Ratings of the Reasons for Schedule Overruns	89
8	Summary of Answers to the Research Questions	114
9	Summary of Findings that Pertain to Appendices E, F and G	116

LIST OF APPENDICES

Appendix	Description	Page
A	Cover Letter	150
B	Questionnaire	151
C	Consent Letter	154
D	Coding Scheme	155
E	Spearman Rank Order Correlation Among the Importance of Reasons for Schedule Overruns	158
F	Spearman Rank Order Correlation Between the Importance of Reasons for Schedule Overruns and Size of Project Relative to the Size of Other Projects Normally Undertaken by the Company	159
G	Spearman Rank Order Correlation Between the Consequences of Schedule Overruns and the Importance of Reasons for Schedule Overruns	160

CHAPTER 1

INTRODUCTION

1.1 RESEARCH QUESTIONS

Schedule overruns are prevalent in software development projects (Jenkins, Naumann and Wetherbe 1984). A United States congressional study identified schedule overruns in software development to be a frequently occurring problem (United States House of Representatives Report 1989). Paulish and Carleton (1994) found that the average schedule overrun for software development projects was 17% of estimated duration with a range of 13% to 42%. Schedule overruns of one to two hundred percent are common (Simmons, Ellis and Escamilla 1993). The average software development project overruns its schedule by 50% and larger projects experience even larger schedule overruns (Gibbs 1994).

Schedule overruns is an important problem faced by organizations that develop software (Brooks Jr. 1975). Schedule overruns result in missed market opportunities (Carmel and Becker 1995); cancellation of contracts (The Ottawa Citizen 1995a); negative publicity (The Ottawa Citizen 1995b and 1995c); lower profits (Fortune 1989, Gibbs 1994); increased costs (Pressman 1982) and

delays in other dependent projects (van Genuchten 1991). Development duration is an important management parameter for controlling software development projects (Andersen 1990). There is a need to deal with schedule overruns in software development projects.

The objective of this study is to examine the relationship between the reasons for schedule overruns and software complexity. A software development project is said to have a schedule overrun when the actual duration of the project exceeds the estimated duration. In this thesis, the duration of a software development project was defined as the calendar time, in weeks, from the start of high level design to first customer delivery (Paulish and Carleton 1994).

This thesis answers the following research questions:

1. What are the most important reasons for schedule overruns in software development projects and what is their relative ranking of importance?
2. Is there a relationship between the importance of a reason for schedule overruns and software complexity measures? Does the importance of a reason for schedule overruns increase or decrease as the software complexity increases?
3. Is there a relationship between the magnitude of schedule overruns and software complexity measures? Does the magnitude of schedule overruns increase or decrease as the software complexity increases?

4. Is there a relationship between the importance of a reason for schedule overruns and the magnitude of schedule overruns? Does the importance of a reason for schedule overruns increase or decrease as the magnitude of schedule overruns increases?
5. Is there a relationship between the relative size of a project and the magnitude of schedule overruns? Do projects that are large or very large in size, compared to projects normally undertaken by the software development organization, experience schedule overruns of greater magnitude?
6. What are the most significant consequences of schedule overruns?

Number of Source Lines of Code (SLOC), the number of modules in the architecture of the software product and the average SLOC per module are used as proxies for software complexity. SLOC refer to the number of lines of executable code and the comment lines contained within it (Basili and Perricone 1984).

This study is both descriptive as well as prescriptive. It is descriptive in that the findings reflect the opinions of project leaders and senior software engineers about the reasons for schedule overruns in software projects. It is prescriptive in that it will suggest corrective action to reduce schedule overruns.

1.2 ORGANIZATION OF RESEARCH QUESTIONS

The model shown in Figure 1 is a representation of the organization of the six research questions. It has five components: software complexity, the most important reasons for schedule overruns and importance of reasons for schedule overruns, size of project relative to other projects normally undertaken by the company, magnitude of schedule overruns, and significant consequences of schedule overruns.

To answer what are the most important reasons for schedule overruns (RQ1), the reasons for schedule overruns have to be identified and ranked according to the importance of their contribution to schedule overruns. To understand the relationship between the importance of a reason for schedule overruns and software complexity (RQ2), and magnitude of schedule overruns (RQ4), the importance rating of the reason has to be correlated with the measures of software complexity and magnitude of schedule overruns respectively. The relationship between software complexity and the magnitude of schedule overruns (RQ3) can be determined by correlating the measures of software complexity and the magnitude of schedule overruns. The relationship between the size of a project relative to the size of projects normally undertaken

Figure 1

Organization of Research Questions

Note: The arrows indicate causality that is assumed in this study. The lines indicate associations that will be investigated in this research.

RQ1: What are the most important reasons for schedule overruns in software development projects and what is their relative ranking of importance?

RQ2: Is there a relationship between the importance of a reason for schedule overruns and software complexity measures? Does the importance of a reason for schedule overruns increase or decrease as software complexity increases?

RQ3: Is there a relationship between the magnitude of schedule overruns and software complexity measures? Does the magnitude of schedule overruns increase or decrease as software complexity increases?

RQ4: Is there a relationship between the importance of a reason for schedule overruns and magnitude of schedule overruns? Does the importance of a reason for schedule overruns increase or decrease as the magnitude of schedule overruns increases?

RQ5: Is there a relationship between the relative size of a project and the magnitude of schedule overruns? Do projects that are larger in size, compared to projects normally undertaken by the software development organization, experience schedule overruns of greater magnitude?

RQ6: What are the most significant consequences of schedule overruns?

by the company and the magnitude of schedule overruns (RQ5) is obtained by correlating these two variables. Finally, to answer (RQ6), the most significant consequences of schedule overruns, the consequences of overruns must be identified and ranked in order of importance.

1.3 COMPARISON OF THIS RESEARCH WITH EXISTING RESEARCH STREAMS

Table 1 positions this research in the context of existing studies in the product development literature. Brown and Eisenhardt (1995) reviewed the product development literature and organized it into three streams: rational plan, communication web and disciplined problem solving. This research shares the following similarities with the stream labeled disciplined problem solving:

1. the key idea - to improve the software development cycle time by identifying and dealing with the causes of schedule overruns
2. measure of performance - shorter software development cycle time.

This research is similar to the stream labeled rational plan in that it is exploratory and collects data from a single well-placed informant on a number of variables using a questionnaire.

This research shares no similarities with the studies in the communication web stream.

Table 1
Comparison Between the Research in This Thesis and Past Research Streams in Product Development

Concepts	Rational Plan	Communication Web	Disciplined Problem Solving	This thesis
Key idea	Success via superior product, attractive market, rational organization	Success via internal and external communication	Success via problem solving	Success via problem solving
Theory	Mostly atheoretical	Information and resource	Information including problem solving	Exploratory
Informants	Single	Multiple	Multiple	Single
Variables	Many independent variables	Many independent variables		Limited number of independent variables
Statistical Methods	Bivariate analysis	Deductive and inductive	Progression from inductive to deductive, single industry, global study	Bivariate analysis
Project leaders		Politician and small group manager	Heavyweight leader	Project leader and senior software engineers
Performance (dependent variable)	Financial success (profits, sales, market share)	Perceptual success (team and management ratings)	Operational success (speed, productivity)	Schedule overruns
Selected studies	Cooper and Kleinschmidt (1987); Stanford Innovation project (Maidique and Zirger 1984, 1985) and Zirger and Maidique (1990)	Ancona and Caldwell (1990, 1992a, 1992b); Dougherty (1990, 1992), Dougherty and Corse (In press)	Harvard Auto Study (Clark, Chew and Fujimoto 1987; Clark and Fujimoto 1991; Hayes et al., 1988), Iansiti (1992, 1993), Eisenhardt and Tabrizi (1995)	

SOURCE: Adapted from Brown and Eisenhardt (1995), Table 1. pg. 347

1.4 CONTRIBUTION

This research is the first to examine the relationship between schedule overruns and software complexity in software development projects. This research prioritizes the reasons for schedule overruns, investigates the effect of software complexity on the importance of a reason for schedule overruns and explores the distribution of schedule overruns across projects that differ in complexity.

Results of this research will assist project leaders and senior software engineers in the allocation of resources and prioritization of activities to reduce or eliminate schedule overruns.

The discussion of the results of this research provides input to process improvement initiatives. Recommendations are made to improve the accuracy of estimates and the efficiency of the development process. This research also contributes ideas for tools to deal with the increase in software complexity.

Finally, this is the first empirical study on the reasons for schedule overruns in software development projects that uses data on Canadian companies. Two of the three previous studies on schedule overruns in software projects used data from companies in the United States (Jenkins, Naumann and Wetherbe 1984, Phan, Vogel and Nunamaker 1988), and the third used data from a Dutch company (van Genuchten 1991).

1.5 ORGANIZATION OF THE THESIS

Chapter 2 provides the literature review. It is divided into three sections. Section 2.1 focuses on the reasons for schedule overruns in development projects. Section 2.2 focuses on software complexity. Section 2.3 summarizes pertinent research papers that were reviewed in the three preceding sections.

Chapter 3, Research Design, describes the data that were collected to answer the research questions as well as how these data were collected. Section 3.1 outlines the research questions. Section 3.2 describes the selection of the sample, Section 3.3 defines the unit of analysis, Section 3.4 identifies the respondents, and Section 3.5 describes the data collected. Section 3.6 describes the data collection method.

Chapter 4 explains how the data were analyzed. Section 4.1 describes the coding and data entry procedures. The computations required to analyze the data and the answers to the research questions are explained in Sections 4.2 to 4.10. Section 4.11 explains how the responses to question 3.4 will be presented. Question 3.4 is "If you had to do this project again, what would you do differently to reduce or eliminate the schedule overruns?"

The results of the data analysis are presented in Chapter 5. This chapter is divided into ten sections. Section 5.1 describes the characteristics of the sample projects used in the study. Sections 5.2 to 5.7 present the results

required to answer the research questions. Section 5.8 summarizes the responses of the respondents to question 3.4 in the questionnaire. Section 5.9 presents other relationships of interest observed in the data and Section 5.10 presents the main findings.

Chapter 6 provides the discussion of the results of this study. Section 6.1 presents a summary of the results of this research. Sections 6.2 to 6.8 discuss the salient results of the study.

Chapter 7 provides the conclusion to the study. It includes recommendations to managers, suggestions for further research, and the limitations of the study.

CHAPTER 2

LITERATURE REVIEW

This chapter is organized into three sections. Section 2.1 examines the project management and software management literature to identify the reasons for schedule overruns. Section 2.2 focuses on software complexity. It defines software complexity, describes the software complexity measures, discusses software complexity measurement, and the effect of software complexity on maintenance costs, programmer productivity and effort. Section 2.3 summarizes nine research papers that were reviewed in the preceding sections.

2.1 REASONS FOR SCHEDULE OVERRUNS

This section focuses on the reasons for schedule overruns in projects. The implicit assumption in this thesis is that improving performance of software development projects, measured by cycle time, begins with adhering to schedule estimates.

The literature on product development and software development was examined. Five studies that directly address overruns in projects were identified and form the core of the literature review. The five studies are: Thamhain and Wilemon (1986), Gupta and Wilemon (1990), Jenkins, Naumann and Wetherbe

(1984), Phan, Vogel and Nunamaker (1988) and van Genuchten (1991). Of these, the first two studies were from the product development literature, not specific to software development projects. The latter three are specific to software development. The sections that follow examine the reasons for schedule overruns identified from these five studies.

Changes in requirements

Change in development objectives or product requirements has been identified as a reason for project schedule overruns. Change in requirements initiated by the customer and management was ranked by project leaders as the number one reason for schedule and budget overruns in the Thamhain and Wilemon (1986) study. Major and minor changes in requirements was one of the reasons for schedule overruns in the Phan, Vogel and Nunamaker (1988) survey of MIS professionals. Poor definition of product requirements was the most important reason for schedule overruns in the study by Gupta and Wilemon (1990).

Under ideal circumstances, product development begins once the product requirements are defined, understood and frozen. In practice this is seldom the case. The two reasons why product requirements are poorly defined are: 1) the customers do not know what they want or cannot articulate their needs in a manner that can be understood by the design engineers (Workman Jr. 1995,

Holtzblatt and Beyer 1993) and 2) the initial requirements analysis did not sufficiently identify and define all customer requirements (Jenkins, Naumann and Wetherbe 1984). In both cases, product requirements are bound to change during the development process.

Senior management involvement and support

Lack of senior management support (Gupta and Wilemon 1990) and insufficient upper management involvement (Thamhain and Wilemon 1986) have been cited as contributing to schedule overruns. Lack of senior management commitment lowers the priority of the product development project. Such projects get mired in conflicts that are seldom resolved quickly. With no top management interest to sustain the development process, development tasks get sidelined. This causes schedule overruns.

Allocation of resources

Lack of resources (Gupta and Wilemon 1990), reduced funding (Jenkins, Naumann and Wetherbe 1984), capacity not available (van Genuchten 1991) and under - staffing (Phan, Vogel and Nunamaker 1988) have been found to cause schedule overruns. The root cause for these reasons is inadequate resource allocation.

It is important that adequate resources be allocated to development projects. In most organizations, projects continually compete for limited human and technical resources with other on-going projects (Adler, Mandelbaum, Nguyen and Schwerer 1995). Yet project leaders often make the wrong assumption that staff and other resources will be available when needed (Abdel-Hamid, Sengupta and Hardebeck 1994). Without adequate resources a project may not proceed as planned and may take longer than estimated to complete.

Project management

Poor project management was cited as a reason for schedule overruns (Gupta and Wilemon 1990). In the absence of a project control system to monitor the status of work, there is no method of keeping track of the progress that has been made. Lack of adequate supervision and control of project resources coupled with poor project monitoring cause schedule overruns.

Planning and estimation

Over-optimistic planning (Phan, Vogel and Nunamaker 1988), unrealistic project plans, under-estimation of the scope of the project, (Thamhain and Wilemon 1986) and under-estimation of the complexity of the application (van Genuchten 1991) result in incorrect estimation of development time.

Accurate estimation of development time is possible only in organizations that exhibit a good understanding of their processes. According to the surveys conducted by the Software Engineering Institute at Carnegie-Mellon University, approximately 81% of the organizations studied were still in Level 1 of the Capability Maturity Model (Humphrey 1988). Level 1 organizations lack a good understanding of their software development process. They do not have an established procedure for estimating development time. Projects experience schedule overruns because estimation procedures are inaccurate. Often, optimistic estimates are provided to fulfill management expectations or in order to secure a contract.

Project teams

Project teams require people from diverse functions and backgrounds (Eisenhardt and Tabrizi 1995, Phan, Vogel and Nunamaker 1988). Team performance depends on the ability of the members to work together and share a common commitment (Ancona and Caldwell 1992a, Thamhain and Wilemon 1986). Members of functionally diverse teams felt that their performance was better than that of less diverse teams (Ancona and Caldwell 1992a). Software development requires the commitment of analysts, designers, programmers, testers, and documentation specialists. Organizing a good project team that includes all the required skills and expertise is difficult (Thamhain and Wilemon

1986). If the project team lacks one or more of the required skills or knowledge, performance will suffer, progress will not be as planned, resulting in schedule overruns.

Problem detection

Inability to detect problems early was identified as one of the top five reasons for schedule overruns by Thamhain and Wilemon (1986). Errors discovered late in the development process require more time and effort to correct than errors discovered earlier (Aggarwal 1993). If the errors are detected early in the development cycle, it takes less time to correct them (Rowen 1990).

Inter-departmental cooperation and communication

The completion of development projects requires the fulfillment of agreements by entities external to the development project team (van Genuchten 1991). Cooperation between departments has been found to be an important factor for the success of development projects (Clark and Fujimoto 1991b, Zirger and Maidique 1990 and Nobeoka and Cusumano 1992). For example, if the marketing department fails to provide clearly defined requirements on time the project cannot be executed according to plan and will experience schedule overruns (van Genuchten 1991). On the other, hand inter-departmental cooperation will help coordinate resource use with other projects

•

and reduce schedule overruns (Abdel-Hamid, Sengupta and Hardebeck 1994, and Phan, Vogel and Nunamaker 1988).

Customer cooperation

Phan, Vogel and Nunamaker (1988) found that inflated user expectation and lack of user cooperation was a major reason for schedule overruns. Customers play an important role in defining the requirements and setting reasonable expectations for the development team. Lack of customer cooperation and inflated expectations frustrates the development team and performance is affected resulting in schedule overruns.

2.1.1 Summary of the reasons for schedule overruns in product development projects

Table 2 shows the reasons for schedule overruns in development projects identified in the literature. Reasons deemed to be similar have been grouped together. In Table 2, the study that mentioned a reason for schedule overruns is marked by an 'X'. Not all studies cited in the literature review are represented in Table 2. Only the five studies mentioned at the beginning of this section were included in this table. This table was used to design the questionnaire used in this study.

Table 2
Reasons for Schedule Overruns Identified in the Literature

Reasons for Schedule Overruns	Product Development Literature		Software Development Literature		
	Gupta and Wilemon (1990)	Thamhain and Wilemon (1986)	Jenkins, Naumann and Wetherbe (1984)	Phan, Vogel and Nunamaker (1988)	van Genuchten (1991)
Poor definition of product requirements, customer and management initiated changes, major and minor changes during development	X	X		X	
Technological uncertainty, technical complexities	X	X			
Lack of senior management support, insufficient upper management involvement	X	X			
Lack of resources, reduced funding levels, capacity not available, under staffing	X		X	X	X
Poor project management	X				

Note: An 'X' indicates that the reason for schedule overruns was identified by the study.

Table 2 (continued)

Reasons for Schedule Overruns	Product Development Literature		Software Development Literature		
	Gupta and Wilemon (1990)	Thamhain and Wilemon (1986)	Jenkins, Naumann and Wetherbe (1984)	Phan, Vogel and Nunamaker (1988)	van Genuchten (1991)
Unrealistic project plan, over-optimistic planning, complexity of application under-estimated, project scope under-estimated, insufficient front end planning, insufficient contingency planning		X		X	X
Staffing problems, problems with organizing project teams		X			
Inability to detect problems early		X			
Lack of inter-departmental cooperation, communication problems				X	X
Conflict and confusion, weak project leadership		X			
Lack of user cooperation and inflated user expectation				X	

2.2 SOFTWARE COMPLEXITY

This section presents definitions of software complexity, describes measures of complexity and software complexity measurement, and examines the relationship between software complexity and maintenance costs, programmer productivity and effort. The section was used to identify the complexity measures used in this thesis.

2.2.1 Definitions of software complexity

Software complexity leads to schedule overruns (Brooks Jr. 1986 and Booch 1994). Basili (1980) defines software complexity as "... a measure of the resources expended by another system while interacting with a piece of software. If the interacting system is people, the measures are concerned with human efforts to comprehend, to maintain, to change, to test etc., that software". Curtis, Sheppard, Milliman, Borst and Love (1979) define complexity as psychological complexity - "as the characteristics of software that make it difficult to understand and work with".

In the above two definitions, software complexity refers to difficulty in interacting with or understanding the code that was produced. Complexity makes it difficult for any one person to understand the development task in its entirety (Booch 1994). The inability to understand complexity results in underestimation

of the scope of the project (van Genuchten 1991). These in turn lead to, among other problems, errors in estimation and schedule overruns (Booch 1994).

2.2.2 Measures of software complexity

Several metrics have been proposed to measure complexity of software. These metrics provide only a crude index of software complexity (Kearney, Sedlmeyer, Thompson, Gray and Adler 1986). The consensus is that there is no single best metric of software complexity (Basili and Hutchens 1983, Conte, Dunsmore and Shen 1986 and Munson and Koshgoftaar 1989). The complexity measures discussed in the literature are described in this section.

Number of SLOC

Software complexity is measured by the number of SLOC. The number of SLOC is the most simplistic measure of complexity (Kearney, Sedlmeyer, Thompson, Gray and Adler 1986). It is, however, strongly related to the sophisticated measures of software complexity such as Halstead's measure and McCabe's cyclomatic number (Curtis, Sheppard and Milliman 1979, Kearney, Sedlmeyer, Thompson, Gray and Adler 1986, and Lind and Vairavan 1989). Programs with a large number of SLOC are also highly complex programs. According to Simmons (1992) complexity increases as the size of the system increases.

Number of modules

A module is defined as a named subfunction, subroutine or the main program of the software system (Basili and Perricone 1984). Typically, a module will perform a single logical task or set of tasks (Banker, Datar, Kemerer and Zweig 1993). Complexity of programming increases as the number of modules increases, (Boydston 1984 and Banker, Datar, Kemerer and Zweig 1993). The increase in complexity is attributed to the difficulty in having to build a large number of interfaces between the modules. The larger the number of modules, the larger the number of interfaces between modules that are required and the greater the complexity.

Average number of SLOC per module

The average number of SLOC per module is the ratio between the number of SLOC and the number of modules in the program. This has also been referred to as the size of modules by Banker, Datar, Kemerer and Zweig (1993). Complexity of programming increases as the lines of code per module increases, (Boydston 1984, and Banker, Datar, Kemerer and Zweig 1993).

Kearney, Sedlmeyer, Thompson, Gray and Adler (1986) describe three metrics of software complexity. The discussion of these three metrics is that provided in Kearney, Sedlmeyer, Thompson, Gray and Adler (1986).

Halstead's measure of software complexity

Halstead's measure of software complexity is a function of the number of operators and operands in the program. The major components of this measure are:

h_1 = number of unique operators

h_2 = number of unique operands

N_1 = Total number of operators

N_2 = Total number of operands

E = Effort required to implement program

Volume of program - ' V ' is given by the formula

$$V = (N_1 + N_2) * \log_2 (h_1 + h_2)$$

Program difficulty - ' D ' is given by the formula

$$D = (h_1 * N_2) / 2h_2$$

Halstead's measure has been extensively used to estimate effort required to implement a program. The estimated effort required to implement the program is given by the formula:

$$E = D * V.$$

McCabe's cyclomatic number

McCabe's cyclomatic number represents the number of linearly independent execution paths through the program. For a well structured module

the cyclomatic number is simply one plus the number of branching statements in the module.

The cyclomatic number is strongly related to Halstead's measure.

Henry and Kafura's measure

Henry and Kafura define complexity of a procedure as follows:

Complexity of a procedure =

length of procedure * (flow of information into procedure * flow of information out of procedures)²

This measure is presented as one that is sensitive to the structural decomposition of the program into procedures and functions. This measure severely penalizes single procedures with a large number of interconnections. This was to encourage modularization. A low complexity score may be achieved by writing a monolithic program with no procedures and hence no flow of information between them.

2.2.3 Software complexity measurement

According to Kearney, Sedlmeyer, Thompson, Gray and Adler (1986) software complexity measures have not been developed with the end use in mind. Software complexity measures have been developed and used without adequate understanding of what is being measured and why it is being

measured. Kearney, Sedlmeyer, Thompson, Gray and Adler (1986) contend that often highly stringent complexity measures are unnecessary. For instance, where software complexity measures are used as descriptive tools in such applications as manpower allocation and not in evaluating the quality of the software, a mere association of the measure with complexity is adequate. In this thesis software complexity is used to understand schedule overruns and not to evaluate the quality of the software.

2.2.4 Software complexity and development duration

None of the studies in the literature investigate the impact of software complexity on development duration or adherence to schedule. Relationships between complexity and maintenance costs, programmer performance and development effort have been studied (Banker, Datar, Kemerer and Zweig 1993, Curtis, Sheppard, Milliman, Borst and Love 1979, Curtis, Sheppard and Milliman 1979, and Lind and Vairavan 1989). The direction of the relationships however, varied widely across studies. It is difficult to hypothesize the nature of the relationship between schedule overruns and software complexity measures based on these studies.

2.3 SUMMARY OF SELECTED ARTICLES

In this section the selected articles that formed the core of the literature review have been summarized. This section is divided into three sub-sections. Sub-sections 2.3.1 deals with articles on product development projects, 2.3.2 summarizes articles on software development projects and 2.3.3 deals with complexity, its measures and effects. Each summary has been organized so as to highlight the objective of the study, the relevant research design, the major findings and the important conclusions of the study.

2.3.1 Studies on product development projects

Two articles, one by Gupta and Wilemon (1990) and the other by Thamhain and Wilemon (1986) are reviewed in this section.

Gupta and Wilemon (1990)

Objective

The objective of this study was to find out why schedule overruns occur in product development projects, the nature of the overruns and what can be done to avoid them. The following issues were addressed:

1. reasons for accelerating the product development process
2. reasons for development delays
3. actions that can help accelerate the development process

4. major concerns expressed by the team members in the product development process

Research method

Data were obtained through a mailed survey and field interviews. Judgment or purposive sampling was used to select the sample. Eighty managers and technologists involved in product development in 12 large technology - based firms filled out the questionnaire. Thirty eight of the 80 individuals who filled out the questionnaire were interviewed.

Percentages were calculated for the number of respondents who cited each reason for schedule overruns.

Findings

Table 3 provides the reasons for schedule overruns in product development identified by Gupta and Wilemon. Of the 38 respondents who were interviewed, 71% identified poor definition of product requirements as a reason for product development delays. In addition to the data shown in Table 3, the authors found that 87% of the managers who were interviewed were of the opinion that the reasons which caused schedule overruns in the past continue to cause schedule overruns in the development projects that were being undertaken at the time of the study.

Table 3

Reasons for Product Development Delays - Gupta and Wilemon (1990)

Percentage of respondents who cited the reason for product development delays are given in parentheses. The percentages are rounded and based on interviews with 38 participants.

Reason for product development delays	
1	Poor definition of product requirements (71%)
2	Technological uncertainty (58%)
3	Lack of senior management support (42%)
4	Lack of resources (42%)
5	Poor project management (29%)
6	Other* (20%)

* The category 'Other' included manufacturing limitations, delays in obtaining patents, tooling problems and problems with vendors.

SOURCE: Gupta and Wilemon (1990). Figure 1. pg. 29.

Conclusions

The authors concluded that there is a strong need to change the processes used to develop products. They made the following recommendations as key factors to successful product development:

1. senior management must give higher priority to product development efforts and quickly resolve conflicts about the allocation of resources,
2. R&D, marketing, engineering and manufacturing functions must be integrated and customers, suppliers and distributors must be involved throughout product development,
3. resources must be made available and managed effectively,
4. an organizational environment that supports teamwork in product development must exist, and
5. product champions must replace weak matrix management structures.

Thamhain and Wilemon (1986)

Objective

The objective of this study was to investigate the types of project control methods used by project managers, the problems they face, project management practices, and performance and criteria for effective project control. The study focused on engineering projects.

Research Method

Data for this study were collected over a three year period. A questionnaire was used to collect data. The sample consisted of 304 general managers and project leaders of 183 projects in electronics, petrochemical, construction and pharmaceutical industries. The respondents were participants in project management workshops and seminars conducted by the authors. The frequency with which a reason caused schedule overruns was ranked on a five point Likert scale. The authors compared the opinions of the project leaders on the importance of the reasons for cost and lead time overruns with those of the general managers.

Kruskal-Wallis analysis of variance by ranks, was used to test the extent of agreement between project leaders' and general managers' opinions of the reasons for schedule overruns. The following test statistics were used to decide the extent of agreement or disagreement:

Strong agreement : If acceptable at > 99% confidence

Agreement : If acceptable at > 90% confidence

Weak Agreement : If acceptable at >80% confidence

Disagreement : If rejected at 80% confidence

Findings

Table 4 provides the results of the study by Thamhain and Wilemon (1986). Project managers and general managers agreed that an unrealistic project plan and inability to detect problems early were two important reasons for schedule overruns.

General managers identified the following five reasons for schedule overruns: 1) insufficient front end planning, 2) unrealistic project plan, 3) project scope under-estimated, 4) customer and management initiated changes and 5) insufficient contingency planning. The top five reasons for schedule overruns according to project leaders were 1) customer and management initiated changes, 2) technical complexities, 3) unrealistic project plan, 4) staffing problems and 5) inability to detect problems early.

The respondents had a long list of root causes for the problems identified in Table 4. These were classified into five categories:

1. problems with organizing project teams
2. weak project leadership
3. communication problems
4. conflict and confusion
- insufficient upper management involvement

Table 4
Reasons for Schedule Slips and Budget Overruns -
Thamhain and Wilemon (1986)

Reason or problem	Rank by		Agreement between Project and General Managers
	Project managers	General managers	
Customer and management initiated changes	1	4	Disagree
Technical complexities	2	10	Disagree
Unrealistic project plan	3	2	Strongly agree
Staffing problems	4	9	Disagree
Inability to detect problems early	5	7	Agree
Priority shifts	6	11	Disagree
Sinking team spirit	7	14	Disagree
Project scope underestimated	8	3	Disagree
Insufficient number of check points	9	8	Agree
No commitment by personnel to plan	10	12	Agree
Insufficient front end planning	10	1	Disagree
Uncooperative support groups	12	13	Agree
Inability to track progress	13	6	Disagree
Insufficient contingency planning	14	5	Disagree
Unqualified project personnel	15	15	Agree

SOURCE: Thamhain and Wilemon (1986). Figure 1. pg. 77.

Note: The differences between the rankings of the project managers and general managers could be attributed to 'self-serving bias' (Robbins 1993). Positive outcomes are attributed to internal factors and negative outcomes are attributed to external factors.

Conclusions

According to the authors, effective control of engineering projects requires the participants to:

- 1. work out a detailed project plan, that involves all key personnel**
- 2. reach agreement on the plan among the project team members and the customer/sponsor**
- 3. obtain commitment from project team members**
- 4. obtain commitment from management**
- 5. define measurable milestones**
- 6. attract and hold quality people**
- 7. establish a controlling authority for each work package**
- 8. detect problems early**

2.3.2 Studies on software development projects

The three studies reviewed in this section are those by Jenkins, Naumann and Wetherbe (1984), Phan, Vogel and Nunamaker (1988) and van Genuchten (1991).

Jenkins, Naumann and Wetherbe (1984)

Objective

The objective of this exploratory study was to provide answers to the following questions:

1. What was a typical systems development project?
2. What were the problems such as cost and schedule overruns facing such projects?
3. What was the extent of use of structured development methodologies?
4. Who were the users of these systems?

Research Method

Seventy two information systems development projects in 23 organizations were studied. These projects were undertaken by companies located in Minneapolis- St. Paul, Houston and Indianapolis. The industries these companies represented included publishing, government, education, hard and soft goods manufacturing, financial insurance, computer service, distribution, construction, energy production and utilities.

Detailed interviews were conducted with the project leaders. Data on estimates of cost, effort and time for each stage of software development were collected. The findings were presented based on the calculation of percentages.

Findings

The authors found that the average schedule overrun was 22% over estimate and 14 of the 72 projects (19.4%) took more than twice as long as estimated (200% of estimate) to complete. Schedule overruns were uniformly distributed among large, medium and small projects. The authors observed that reduced funding levels contributed to schedule overruns.

Conclusions

The authors recommended that users should be part of the systems development process, especially when user requirements are being defined.

Techniques used to estimate project costs and personnel requirements need to be constantly improved by comparing actual performance to initial estimates. Since initial estimates are seldom accurate, the organization should be psychologically prepared for significant deviations.

Phan, Vogel and Nunamaker (1988)

Objective

Phan, Vogel and Nunamaker (1988) studied which project management and control methods are effective in software project development projects. They addressed the following questions:

1. Why did development projects exceed budgets?

2. Why did development projects overrun schedules and how did management information systems professionals deal with it?

3. What tools were used to manage development projects?

Research method

Questionnaires were sent to 827 randomly selected members of the American Institution of Certification of Computer Professionals. A hundred and ninety one completed questionnaires were returned (response rate 23%). The respondents included leaders/ managers, programmers, analysts and consultants, and top executives. Percentages were calculated to determine the frequency of schedule overruns, the reasons for schedule overruns, and the most frequently used methods of dealing with schedule overruns.

Findings

The authors found that 17.5% of software projects in the sample were rarely or never delivered late, while 32.9% were always or usually late. The main reasons for schedule overruns were over-optimistic planning, changes after a major portion of development had been completed, lack of user cooperation and inflated user expectation, lack of inter-departmental cooperation, and understaffing.

The two most common and effective methods of dealing with schedule overruns were to upgrade the priority of the project and to pay for overtime. Adding manpower was not an effective method of reducing schedule overruns.

The following positively influenced the delivery of software on time:

- 1. project manager possessed good management skills**
- 2. project costs were paid by the requesters or users**
- 3. team members and leaders were specialists**
- 4. feedback requiring major changes were not considered after design was frozen.**

Conclusions

The authors made the following recommendations to ensure that user expectations are met without excessive schedule overruns:

- 1. emphasize managerial skills in the selection of team leaders**
- 2. state problem definition, opportunities, environmental objectives, goals and objectives clearly and explicitly**
- 3. use a steering committee and joint staff to reduce bureaucracy and lack of interdepartmental cooperation**
- 4. recruit team staff from multiple backgrounds and strive for a balance of specialists and generalists**
- 5. provide feedback immediately**

6. use project management tools such as PERT/CPM and on-line monitoring
7. upgrade priorities and arrange for staff overtime

van Genuchten (1991)

Objective

This study examined the reasons for differences between plans and reality in software development activities. The intent was to recommend actions to improve the development process. The study aimed to answer the following questions:

1. What were the predominant reasons for delay in the activities studied?
2. What was the distribution of reasons for delay between departments?
3. How was the delay distributed over the phases of a project?

What actions for improvement could prevent delay in future projects?

Research method

Data on reasons for delays in 160 activities were collected from six development projects in one software development department. Data on the differences between plans and reality for development effort and duration were collected. Duration of an activity was defined as the calendar period, in weeks, between the starting and ending dates. Reasons for delays were classified into the six groups shown in Table 5. The reason for a difference between plan and reality were assigned by the project leader in consultation with the researcher.

Table 5
Classification of Reasons for Effort and Schedule Overruns-
van Genuchten (1991)

CAPACITY RELATED REASONS 11 Capacity not available because of overrun in previous activity 12 Capacity not available because of overrun in other activity 13 Capacity not available because of unplanned maintenance 14 Capacity not available because of unplanned demonstration 15 Capacity not available because of other unplanned activities 16 Capacity not available because of other causes 19 Other
PERSONNEL-RELATED REASONS 21 Too little experience with development environment 22 More inexperienced people in team than expected 29 Other
INPUT-REQUIREMENTS NOT FULFILLED 31 Requirements late 32 Requirements of insufficient quality 33 (Specs. of) delivered software late 34 (Specs. of) delivered software of insufficient quality 35 (Specs. of) hardware late 36 (Specs. of) delivered hardware of insufficient quality 39 Other
PRODUCT RELATED REASONS 41 Changing requirements during activity 42 Changing of the interfaces during the activity 43 Complexity of the application under-estimated 44 More problems than expected with performance requirements or memory constraints 45 Product of insufficient quality developed (redesign necessary) 49 Other
ORGANIZATION RELATED REASONS 51 Less continuity in project staffing than expected 52 More interruptions than expected 53 Influence of software Quality Assurance 54 Bureaucracy 50 Other
TOOLS RELATED REASONS 61 Development tools too late or inadequately available 62 Test tools too late or inadequately available 69 Other

SOURCE: van Genuchten (1991). Table V. pg. 586.

Percentages were calculated for the number of activities that experienced schedule overruns and for the number of times each of the reasons were cited for schedule overruns.

Findings

Thirty percent of the activities were finished according to plan while more than 45% of the activities had a schedule overrun of one week or more. Another significant finding was that the schedule overruns increased in activities that were undertaken towards the final stages of the project. The most important reasons for schedule overruns were capacity related, specifically reasons 12-16 in Table 5. Product related reasons were the second most important cause for schedule overruns. For example, reason 43 in Table 5, complexity of application under-estimated, was given as a reason for delays in 20% of the cases.

The study also found that the reasons for delays varied widely from one department to another.

Conclusions

The author concluded that the execution of a project according to schedule depended on the fulfillment of agreements by entities external to the development team. Since the reasons for delays varied widely, each development department must ascertain the specific reasons for schedule overruns before improvements could be made.

2.3.3 Software complexity

This section summarizes four papers that deal with software complexity measurement, software complexity and maintenance costs, programmer productivity and effort.

2.3.3.1 Software complexity measurement

This section summarizes a paper that underlines the need to use appropriate measures of complexity for evaluating various aspects of software development.

Kearney, Sedlmeyer, Thompson, Gray and Adler (1986)

Objective

The objective of the paper was to emphasize a programming-behavior approach to developing, validating and using software complexity measures. The authors affirm that a majority of the software complexity measures that have been developed are not widely accepted and applied because they were developed with little regard for the programmer, the programming task or the programming environment. The authors make the case for developing software complexity measures that are grounded in the theory of programming behavior.

The term *programming* encapsulates a variety of tasks that programmers perform such as design, coding, debugging, testing, modification, and documentation. The authors provide insights derived from their experience.

Insights

1. The first two steps in developing a measure are: (a) specifying what is being measured and (b) why it is being measured. This description should be supported by programming behavior.
2. The difficulty of the task and programming behavior are influenced by the experience and ability of the programmer, the specific task, the problem domain and the programming environment.
3. The type of experiences that influence the difficulty of the task include general knowledge of programming languages, techniques, and algorithms, and knowledge of the application area.
4. Software complexity metrics should also be developed with the end-use in mind. For example, software metrics are used as descriptive tools in such applications as manpower allocation *and estimation of development duration*. In such cases, a mere association of the measure with complexity is adequate¹.

¹ This clarifies the distinction between program properties that cause a program to be complex and characteristics of a program that are associated with complexity. SLOC, one of the measures used in this thesis, is associated with complexity.

5. The complexity measure should be robust, that is, a reduction in the measure should lead to an improvement in the program; normative, in that one should be able to judge the relative complexity; specific, in that it should be able to identify what contributes to the complexity; and prescriptive, that is it should suggest techniques to avoid excess complexity.
6. Experimental observations in support of a software complexity measure should be used with caution. Most measures are not generalizable to the various programming languages, programming tasks and the algorithms implemented.

2.3.3.2 Software complexity and maintenance costs, programmer productivity and effort

This section summarizes three papers that study the effect of software complexity on maintenance costs, programmer performance, and development effort.

Banker, Datar, Kemerer and Zweig (1993)

Objective

The objective of this study was to estimate the effect of software complexity on the cost of maintenance projects. The authors believe that maintenance projects have much in common with development projects. Thus

the results of the study by Banker, Datar, Kemerer and Zweig (1993) can be generalized to development projects.

Research Method

Figure 2 shows the conceptual model that was tested in the study. The research investigated the relationship between software complexity and maintenance costs while controlling for project factors. Data were collected from a regional bank in the United States. Its systems contained 18 million lines of COBOL code. The following hypotheses were tested using F-tests.

H₁ Controlling for other factors known to affect software maintenance costs, the costs will depend significantly on average procedure size with costs rising for applications whose average procedure size is either very large or very small.

H₂ Controlling for other factors known to affect software maintenance costs, the costs will depend significantly on average module size with costs rising for applications whose average module size is either very large or very small.

H₃ Controlling for other factors known to affect software maintenance costs, software maintenance costs will depend significantly on the density of branching, with costs rising with increases in the incidence of branching.

Figure 2

Software Maintenance Project Costs Conceptual Model

SOURCE: Banker, Datar, Kemerer and Zweig (1993) pg. 87

Findings

- 1. The study confirmed H_1 . There was a 'U'- shaped relationship between procedure size and maintenance costs. The costs of maintenance projects rose as procedure size increased or decreased from an optimum point.**
- 2. The hypothesis that costs increase for both large as well as small number of module (H_2) was not supported. However, the authors felt that had sufficiently large modules been part of the study there was evidence to support the existence of a 'U'-shaped relationship between the number of modules and maintenance costs.**
- 3. Software maintenance costs increased linearly with an increase in branching. This observation confirmed H_3 .**

Conclusions

- 1. Good programming practices such as optimal procedure size, optimal number of modules and lower branching reduce complexity.**
- 2. The regression models developed in this research allows managers to justify investments in Computer Aided Software Engineering tools to improve programming practices by enabling them to estimate benefits in terms of reduced maintenance costs.**
- 3. Increased software complexity decreases the productivity of software maintainers.**

Curtis, Sheppard and Milliman, (1979)

Objective

The objective of this experiment was to test the value of complexity metrics in predicting the time required for a programmer to find an error in the software. The intent was to enable prediction of manpower requirements for maintenance based on the complexity of the software.

Research Method

Fifty-four professional FORTRAN programmers with an average experience of 6.6 years participated in the experiment. To control for individual differences in performance a within-subject 3⁴ factorial design was employed. The experimental task involved finding three different errors in three software programs. These three programs had three types of control flows and varied in length from 25 to 225 lines of code. The permutations and combinations of these four factors resulted in the 3⁴ factorial design. Preliminary tasks were provided to reduce learning effects on the experimental tasks and to provide a basis for comparing the abilities of the participants to perform a task of this nature. The first 27 participants were given the three programs in order of complexity, the second set of 27 participants replicated the conditions exactly except that the order of presentation was different. The maximum time allowed for completion of the preliminary task was 45 minutes and 60 for the experimental task. The actual

time for the bug-detection was measured to the nearest minute. Halstead's *E* as well as McCabe's ν (G) (complexity metrics) as well as the total statements of FORTRAN code were collected. Intercorrelations among complexity metrics and correlations between performance and complexity measures were computed.

Results

Table 6 shows the intercorrelations among the complexity metrics. At the sub-routine level, all three measures tended to measure a common construct of complexity. When the intercorrelations were computed for the total program, however, the correlation between total lines of code and McCabe's ν (G) was large while the correlation between Halstead's *E* and lines of code was smaller.

Table 6

Intercorrelations Among Complexity Measures

Correlations		
Measures	Halstead's <i>E</i>	McCabe's <i>v</i> (G)
Subroutine:		
<i>v</i> (G)	.92***	
Length (SLOC)	.89***	.81***
Program:		
<i>v</i> (G)	.76***	
Length (SLOC)	.56***	.90***

n=27

***p ≤ .001

SOURCE: Curtis, Sheppard and Milliman (1979) pg. 358.

Table 7 shows the correlations between performance and complexity measures. At the subroutine level all three measures predicted performance equally well, at the program level, Halstead's E was the best predictor of performance, followed by McCabe's v (G) and then by the length or SLOC.

A curvilinear relationship was detected between Halstead's E and performance, suggesting that as Halstead's E grew larger, the psychological complexity of the program increased.

Conclusions

1. At the subroutine level all three measures of complexity measure the same construct. At the program level, Halstead's E measures something in addition to that which is measured by length (SLOC).
2. The psychological complexity of small programs is adequately represented by SLOC. When the code grows beyond the subroutine its complexity to the programmer is better assessed by constructs other than SLOC.
3. Halstead's E and McCabe's v (G) are related to the difficulty programmers experience in locating errors in code. As complexity increased, error detection became more difficult.
4. These measures may be used for providing feedback to programmers about the complexity of code they have produced, as well as for allocation of

Table 7

Correlations Between Performance and Complexity Measures

Correlations		
Measures	Subroutine	Program
<i>E</i>	.66***	.75***
<i>v</i> (G)	.63***	.65***
Length	.67***	.52**

n= 27

** $p \leq .01$

*** $p \leq .001$

SOURCE: Curtis, Sheppard and Milliman (1979) pg. 358.

resources for maintenance projects. But more evaluative research needs to be done to assess the validity of these used in ongoing software projects. Code which is complex may also be more error-prone and difficult to test.

Lind and Vairavan (1989)

Objective

The objective of this experimental investigation was to study:

1. the relationship between software complexity metrics and software development effort
2. the relationship among various software complexity metrics

The software complexity metrics studied were Halstead's program length, McCabe's control complexity and program bandwidth. In addition to the above mentioned metrics, data were collected on total SLOC - divided into lines of code and comment lines, total number of characters - divided into code characters and comment characters.

Research Method

A commercial, medical imaging systems software used in real-time application was studied. The software consisted of 4500 routines with a total of about 400,000 lines of code. Of those routines, 58% which were in Pascal and

29% which were in FORTRAN were studied. Pearson correlation coefficient was used to estimate the degree of linear relationship between variables.

Results

The study found:

1. Lines of code had the strongest correlation to development effort followed by McCabe's control complexity and Halstead's program length.
2. A relatively high correlation between the number of comment characters and development effort. A likely explanation for this was that programmers used comments heavily to explain complex code².
3. The number of changes per 100 lines of code (defect density) decreased as the complexity metrics increased up to a certain point. After this point however, the defect density increased as the complexity metric values increased. This questions the validity of earlier conclusions (Shen, Yu, Thebaut and Paulsen 1985) that (a) larger programs were coded with greater care and (b) there was a high possibility of errors going undetected in larger programs. The authors conclude that the substantial increase in defect density after a particular point is because the complexity is excessive for programmers to comprehend.

² Skilled programmers may be able to write a significant part of the program in relatively few lines. The complexity of such a program would be reflected in the amount of comments lines required to explain it. The measure of SLOC in this thesis includes both executable code and comment lines.

4. The degree of correlation between program bandwidth and every other metric was found to be low. This was contrary to an earlier study conducted on building automation software. A possible explanation is that medical imaging software was less nested than building automation software.

Conclusions

Simple measures such as SLOC correlated to development effort just as well or better than many standard software metrics.

Experimental studies on the relationship between software metrics and development effort are required to provide further insights into the development process.

CHAPTER 3

RESEARCH DESIGN

This chapter describes the research design used in this thesis. Section 3.1 states the research questions. Section 3.2 describes the process of sample selection. Sections 3.3 and 3.4 identify the unit of analysis and the characteristics of the respondents. Section 3.5 describes the questionnaire used to collect the data. Section 3.6 describes how the data were collected.

3.1 RESEARCH QUESTIONS

The research questions are:

- RQ1. What are the most important reasons for schedule overruns in software development projects and what is their relative ranking of importance?**
- RQ2. Is there a relationship between the importance of a reason for schedule overruns and software complexity measures? Does the importance of a reason for schedule overruns increase or decrease as software complexity increases?**
- RQ3. Is there a relationship between the magnitude of schedule overruns and software complexity measures? Does the magnitude of schedule overruns increase or decrease as software complexity increases?**

RQ4. Is there a relationship between the importance of a reason for schedule overruns and the magnitude of schedule overruns? Does the importance of a reason for schedule overruns increase or decrease as the magnitude of schedule overruns increases?

RQ5. Is there a relationship between the relative size of a project and the magnitude of schedule overruns? Do projects that are large or very large in size compared to projects normally undertaken by the software development organization experience schedule overruns of greater magnitude?

RQ6. What are the most significant consequences of the schedule overruns?

3.2 SAMPLE SELECTION

3.2.1 Population

The population consists of all software development projects undertaken by the 614 companies listed as software companies in the 1994 edition of the Directory of High Technology Companies in Metro Ottawa published by the Ottawa-Carleton Economic Development Corporation (OCEDCO) which experienced schedule overruns and which were completed after April 30th, 1993. The projects that made up the population delivered custom software,

commercial software, information systems software, critical application software, and embedded software.

3.2.2 Sampling method

Convenience sampling was used to select the sample. Initially, a phone call was made to the number of the company listed in the directory. The researcher identified himself to the person who answered the phone as a graduate student at Carleton University's School of Business collecting data for a thesis titled 'Reasons for Schedule Overruns in Software Development Projects'. In most cases, the person answering the phone was the secretary or the receptionist, but at times engineers answered the phone. In the former case, the researcher asked to speak with either a software project leader or a senior software engineer. If no one was available to speak with the researcher, the consent letter shown in Appendix C was sent to the person identified by the secretary or the receptionist. The consent letter requested the names of project leaders or senior software engineers who could provide data for the study. Questionnaires were sent to all persons identified in the reply to the consent letter. If software project leaders or senior software engineers were available to speak to, their verbal consent was sought and then the questionnaire was either faxed or e-mailed to them. None of the questionnaires was mailed to respondents.

Each respondent was allowed to report on a maximum of two projects. However, none of the respondents reported on more than one project. It was also decided that there would be only one respondent per project. In case there was more than one response per project, the first complete response was to be included in the analysis. However, there were no multiple responses for any project and hence this guideline never had to be applied. To ensure that reasons specific to one development environment did not skew the data, not more than two projects were included from organizations that employ fewer than 50 people. It was assumed that in organizations employing more than 50 people, the development environment differs significantly from project to project. Therefore up to five software development projects were permitted from each of these organizations.

3.2.3 Sample size

The sample size was 41 software development projects.

3.3 UNIT OF ANALYSIS

The unit of analysis was a software development project.

3.4 RESPONDENTS

Respondents were the project leaders and senior software engineers of the software development projects in the sample. They were selected as

respondents because they were knowledgeable about the project and were also responsible for project control (van Genuchten 1991).

3.5 DATA COLLECTED

A two page questionnaire was used to collect the data for this research. A questionnaire survey was within the resources of the researcher and was considered appropriate for the type of data required in this study. The initial version of the questionnaire was pre-tested on ten software engineers. The questionnaire was modified based on the feedback received from ten respondents. These ten respondents were not part of the final sample. In addition, input from two persons who were familiar with software development projects was also included in the questionnaire.

The questionnaire was divided into four sections. Section 1 was for project data, Section 2 was for data on software complexity, Section 3 was for data on schedule overruns, and Section 4 was for the respondent's e-mail address.

3.5.1 Project data

Section 1 of the questionnaire sought the name of the respondent and a contact phone number (question 1.1), the name of the company (question 1.2) and the main line of business of the company (question 1.3). This information

was used to contact respondents only if a questionnaire was returned incomplete or for follow up interviews if needed. Information on the main line of business of the company was used to report the composition of the sample.

The title of the project (question 1.4) was used to ensure that two respondents do not provide data on the same project. The date of project completion (question 1.5) was requested to eliminate those projects that were completed prior to April 30, 1993. The description of the project (question 1.6) was sought for classification purposes and to assist in the interpretation of the data.

3.5.2 Software complexity data

Data on Halstead's measure, McCabe's cyclomatic number and Henry and Kafura's measure are not usually collected by companies in the Ottawa-Carleton area. Calculating these metrics was tedious. The software product for which complexity is being measured has to be evaluated using special software. Thus, data on these three software complexity measures could not be collected. In this study, software complexity metrics were not sought to evaluate the quality of the program, but as descriptive tools. Therefore, measures associated with complexity were thought to be adequate (Kearney, Sedlmeyer, Thompson, Gray and Adler 1986). Moreover, the respondents could relate to the number of

SLOC, number of modules and average number of SLOC per module more readily than to the other measures.

Questions 2.1 and 2.2 collected data on: the number of SLOC, the language in which the code was written, and the number of modules developed for the project.

Data on the number of SLOC and the number of modules are basic project - related data. These data are readily available. The average number of SLOC per module, a ratio of the number of SLOC to the number of modules, was calculated from the above data.

The number of SLOC and the number of modules are good indicators of complexity (Boydston 1984, Banker, Datar, Kemerer and Zweig 1993, Kearney, Sedlmeyer, Thompson, Gray and Adler 1986, Lind and Vairavan 1989 and Simmons 1992). Modules that perform a single logical task are less complex than modules which carry out more than one set of logical tasks. Complexity of programming increases as the average number of SLOC per module increases (Boydston 1984, Banker, Datar, Kemerer and Zweig 1993).

Monolithic programs would be considered low in complexity had just the number of SLOC and the number of modules been used to measure complexity. To overcome this problem the average number of SLOC per module was also

included as a measure of complexity in this thesis. If the average number of SLOC per module is high, the module is unlikely to be devoted to a single purpose. Such modules would therefore be more complex than those that are devoted to a single purpose.

The number of SLOC, the number of modules and the average number of SLOC per module are not absolute measures of complexity. They are relative measures. For example, assuming that all other things are equal, a project that has 200,000 SLOC may be more complex than another with 100,000 SLOC, but the former is not necessarily twice as complex as the latter.

Respondents were requested to compare the size of the software development project with that of other software development projects that were normally undertaken by the company (question 2.3). This was to determine if there was a relationship between relative size of a project and schedule overruns.

3.5.3 Schedule overrun data

Question 3.1 requested data on the estimated and actual duration of the project in weeks. This information was used to calculate the magnitude of schedule overruns. Question 3.2 requested the respondent to indicate the significance of the consequences of the schedule overruns in terms of project cost (Pressman 1982), customer satisfaction (Pressman 1982), progress of

dependent projects (van Genuchten 1991) and financial success (Fortune 1989 and Gibbs 1994). A category 'other' was included for the respondent to indicate consequences other than those listed. The significance was measured on a 5 point Likert scale. On this scale 1 represents 'very significant' and 5 represents 'not at all significant'.

3.5.4 Reasons for schedule overruns included in the questionnaire

Question 3.3 asked the respondent to rank the importance of the reasons for schedule overruns. A five point Likert scale was used for this purpose. One, on this scale, signifies that the reason is 'not at all important' and five means the reason is a 'very important' cause of schedule overruns. A separate category designated as N/A, not applicable, was to be used if the reason did not cause schedule overruns.

Reasons identified in the literature

Table 2 in Chapter 2 identified the reasons for schedule overruns found in the product development and software development literature. Table 8 relates the reasons for schedule overruns used in the questionnaire with the reasons for schedule overruns identified in Table 2. Eleven reasons for schedule overruns were included in the questionnaire. Nine of these 11 reasons for schedule overruns were derived directly from Table 2.

Table 8
Relationship Between the Reasons for Schedule Overruns Found in the Literature and the Reasons Included in the Questionnaire

No.	Reasons for schedule overruns found in the literature	Reasons included in the questionnaire
1	Poor definition of product requirements, customer and management initiated changes, major and minor changes during development	Changes in specifications
2	Technological uncertainty, technical complexities	
3	Lack of senior management support, insufficient upper management involvement	Lack of senior management support
4	Lack of resources, reduced funding levels, capacity not available, under staffing	Inadequate resource allocation
5	Poor project management	Poor project management
6	Unrealistic project plan, over-optimistic planning, complexity of application under-estimated, project scope under-estimated, insufficient front end planning, insufficient contingency planning	Incorrect estimation of development time
7	Staffing problems, problems with organizing project teams	Problems with organizing project teams
8	Inability to detect problems early	Inability to detect problems early
9	Lack of inter-departmental cooperation, communication problems	Lack of inter-departmental cooperation
10	Conflict and confusion, weak project management	
11	Lack of user cooperation and inflated user expectation	Inflated user expectation

These nine reasons are:

- 1. Changes in specifications**
- 2. Lack of senior management support**
- 3. Inadequate resource allocation**
- 4. Poor project management**
- 5. Incorrect estimation of development time**
- 6. Problems with organizing project teams**
- 7. Inability to detect problems early**
- 8. Lack of inter-departmental cooperation**
- 9. Inflated user expectation**

Except for changes in specifications the other reasons for schedule overruns included in the questionnaire are self-explanatory. The use of changes in specifications in the questionnaire requires an explanation. Typically, product specifications are derived from a product requirements document. The product requirements document is a wish-list of functions and attributes to be incorporated in the product being developed. Specifications are tempered by technology, time and resource constraints. Developers have to rely on marketing personnel for specifying customer requirements. Marketing personnel may sometimes become barriers to requirements elicitation (Carmel and Becker 1995). Often a software development team is not familiar with the details of the

requirements and works from the specifications. Thus, in this research, changes in specifications was included as a reason for schedule overruns instead of changes in requirements. Poorly defined product requirements or changes in product requirements made either by the customer or management invariably result in changes in specifications. Frequent changes in specifications distract, confuse and de-motivate the development team (Bennatan 1992). Time and effort are wasted in making adjustments to incorporate the changes. The time and effort required is even greater when changes are made after a major portion of the development has been completed. If specifications are stable, more time is spent on tasks related directly to the development of the product.

Two other reasons thought to be important for schedule overruns, were included in the questionnaire. Both were suggested by two individuals familiar with software development projects who examined the questionnaire. These reasons are:

10. Inadequate techniques and

11. Lack of experience of the team

Inadequate techniques

At times the technique used to develop the product is inadequate and does not meet the requirements of the project. Examples of inadequate techniques are the use of Object Oriented methodology for designing user

interface instead of functional decomposition or using Rumbaugh-Booch methodology for developing real time systems instead of Shlaer-Mellor methodology.

Lack of experience of the team

Team members do not always have adequate experience with the type of product or application being developed. They may not have had sufficient experience in the methodology being employed in the project or may not have had experience in the problem domain.

Experience influences the difficulty of the programming task (Kearney, Sedlmeyer, Thompson, Gray and Adler 1986). Programming skills developed through practice, such as interpreting system diagnostics, are also important determinants of programming behavior. Soloway and Ehrlich (1984) present evidence that experienced programmers develop schemata for stereotypical action sequences in programs. These schemata provide the expert programmer with a framework for understanding the program.

Two reasons that were in Table 2 were excluded from the questionnaire. First, technical complexity was not included as a reason for schedule overruns as this study examines the relationship between the reasons for schedule overruns and complexity. Second, weak project leadership was not included as a reason for schedule overruns in the questionnaire as it was considered

inappropriate to have respondents who happen to be project leaders evaluate the extent to which weak project leadership affected schedule overruns.

Question 3.4

Question 3.4 asked respondents what they would do differently to reduce or eliminate the schedule overruns if they had to do the project again.

3.5.5 Request for a copy of the findings

A space was provided at the end of the questionnaire for respondents to provide their e-mail address should they be interested in receiving a copy of the findings. This was to serve as an incentive for the respondents to return the completed questionnaire.

3.6 DATA COLLECTION METHOD

Respondents identified through the OCEDCO directory were telephoned before the questionnaires were sent out. This was expected to improve the questionnaire response rate. Questionnaires were either faxed or e-mailed to the respondents. None of the questionnaires was mailed. It was thought that faxes and e-mail would elicit a greater number of responses than the more passive mail.

In all 180 telephone calls were made, 47 questionnaires were faxed and 18 questionnaires were e-mailed. Thirty five of the 47 questionnaires that were

faxed were returned. Upon verification 30 of the 35 questionnaires were valid. Of the 18 questionnaires that were e-mailed 12 were returned. Eleven of those were valid and one was rejected. The response rate for the faxes was 74% and that for the e-mails was 66.67%. The combined response rate for the survey was 72.3%.

A log was used to record the names of the respondents contacted, the date they were contacted and the date on which the responses were received. The log was used to follow-up non-responses as well as incomplete responses. The reasons cited for the inability to respond were 1) too busy to spare the time, and 2) complexity data not readily available. All responses were checked for completeness. Incomplete responses were followed-up with the respondent. If adequate clarification was not available the response was set aside.

The questionnaire, including the cover letter, was also posted under the title 'Schedule Overruns Survey' in the following newsgroups on the internet: comp.software-eng; biz.comp.software; compt.std.c; comp.std.c++; comp.lang.ada; comp.lang.apl; comp.lang.c; comp.lang.c++ and comp.software.config-mgmt. The initial post was followed up with another post a week later in comp.software-eng only. The survey was not re-posted on the other newsgroups because I was informed by the users of these newsgroups that they were inappropriate for such surveys. No valid response was received to

these internet postings. There was, however, one offer to participate from an American developer working on a Government of Canada contract. I found these newsgroups on the internet inappropriate for seeking survey responses.

CHAPTER 4

DATA ANALYSIS

This chapter outlines how the data were analyzed in order to answer the research questions. Section 4.1 describes the coding and data entry procedures. The computations required to analyze the data and answer the research questions are explained in Sections 4.2 to 4.10. Section 4.11 explains how the responses to question 3.4 (respondents' recommendations to reduce or eliminate schedule overruns) were analyzed and presented.

4.1 CODING AND DATA ENTRY

The coding scheme provided in Appendix D was used to code and enter the data into a computer. Except for questions 1.6, 2.3, 3.2 and 3.3, Appendix D is similar to the questionnaire in Appendix B. In questions 1.6 and 2.3 the option-boxes have been replaced by numbers that represent the codes.

Responses to question 1.6 were coded as follows: custom software development was coded as `1', embedded software development was coded as `2', software development for a target market was coded as `3'. `Other' was coded as `4'.

Responses to question 2.3 (relative size of projects) were coded from '5' to '1'. '5' represents 'very large' relative size of project and '1' represents 'very small' relative size of project.

The coding of responses to question 3.2 was as follows: 'Very significant' consequence was coded as '5' and 'not at all significant' was coded as '1'.

In question 3.3, when a reason for schedule overruns was marked as 'Not applicable' it was coded as Zero. When a reason is marked 'Not applicable' it is lesser in importance than 'Not at all important'. Since '0' is less than '1', this coding is consistent with the ordinal scale of measurement.

4.2 CALCULATION OF THE AVERAGE NUMBER OF SLOC PER MODULE

The average number of SLOC per module was calculated as the ratio between the number of SLOC and the number of modules in the project.

4.3 CALCULATION OF THE MAGNITUDE OF SCHEDULE OVERRUNS

The magnitude of schedule overruns was calculated by subtracting the estimated time from the actual time. In addition schedule overruns as a percentage of estimated duration and actual duration were also calculated.

4.4 DESCRIPTIVE STATISTICS

Measures of central tendencies such as mean, medians, standard deviation and range were calculated for the following:

1. the number of SLOC (responses to question 2.1),
2. the number of modules (responses to question 2.2),
3. the average number of SLOC per module (calculated from responses to questions 2.1 and 2.2),
4. estimated duration (responses to question 3.1),
5. actual duration (responses to question 3.1), and
6. magnitude of schedule overruns, schedule overruns as a percentage of estimated and actual durations (calculated from responses to question 3.1).

Frequency counts were done on the responses to questions 1.3, 1.6, 2.1.a and 2.3.

4.5 ANSWER TO RESEARCH QUESTION 1

What are the most important reasons for schedule overruns in software development projects and what is their relative ranking of importance?

Ratings on the five-point Likert scale used in question 3.3 reflected the respondents' opinions of the importance of the reasons listed for causing schedule overruns. Responses marked as 'not applicable' were coded as zero and included in the calculation. The mean of the respondent ratings was calculated for each of the reasons for schedule overruns. The reason for schedule overruns with the highest mean rating was identified to be the most

important cause for schedule overruns, the reason with the second highest mean rating was the second most important, and so on.

The non-parametric Friedman two-way Anova of K-related samples test of equality of medians were performed to determine the significance of the differences between the ratings of the reasons for schedule overruns. Further, Wilcoxon matched-pairs signed ranks test was done to confirm the most important reasons for schedule overruns.

4.6 ANSWER TO RESEARCH QUESTION 2

Is there a relationship between the importance of a reason for schedule overruns and software complexity measures? Does the importance of a reason for schedule overruns increase or decrease as software complexity increases?

For each reason for schedule overruns, the respondent's ratings were correlated with each of the three measures of complexity. The non-parametric Spearman rank order correlation coefficient was calculated to determine whether or not a relationship existed between software complexity and the importance of a reason.

A positive correlation indicated that the importance of a reason for schedule overruns increased as the software complexity increased; a negative correlation indicated that the importance of a reason decreased as the software complexity increased. An absence of correlation between complexity and the

importance of a reason for schedule overruns indicated the absence of a relationship between the two.

4.7 ANSWER TO RESEARCH QUESTION 3

Is there a relationship between the magnitude of schedule overruns and software complexity measures? Does the magnitude of schedule overruns increase or decrease as software complexity increases?

The magnitude of schedule overruns, magnitude of schedule overruns expressed as a percentage of estimated duration as well as actual duration were correlated with each of the three measures of complexity. The data for these are on a continuous scale. Hence, Pearson correlation coefficient was used to establish the relationship between the magnitude of schedule overruns and software complexity.

A positive correlation indicated that the magnitude of schedule overruns increased as software complexity increased; a negative correlation indicated that the magnitude of schedule overruns decreased as software complexity increased. No correlation signified that there was no relationship between complexity and the magnitude of schedule overruns.

4.8 ANSWER TO RESEARCH QUESTION 4

Is there a relationship between the importance of a reason for schedule overruns and magnitude of schedule overruns? Does the importance of a reason for schedule overruns increase or decrease as the magnitude of schedule overruns increases?

Respondent ratings for each reason for schedule overruns were correlated with the magnitude of schedule overruns. Spearman rank order correlation coefficient was used to establish the relationship between the two variables.

A positive correlation indicated that as the magnitude of schedule overruns increased, the importance of a reason for schedule overruns also increased. A negative correlation indicated that as the magnitude of schedule overruns increased the importance of a reason for schedule overruns decreased. No correlation signifies the absence of relationship between the two variables.

4.9 ANSWER TO RESEARCH QUESTION 5

Is there a relationship between the relative size of a project and the magnitude of schedule overruns? Do projects that are larger in size compared to projects normally undertaken by the organization experience schedule overruns of greater magnitude?

The magnitude of delays were correlated to the relative size of the project. Relative size of a project is the size of the software development project compared to other software development projects normally undertaken by the company. Spearman rank order correlation coefficient was used to establish the relationship between the magnitude of schedule overruns and the relative size of project.

A positive correlation between the magnitude of schedule overruns and the relative size of the project indicated that the magnitude of schedule overruns increased as the relative size of the project increased. A negative correlation indicated that the magnitude of schedule overruns decreased as the relative size of the project decreased. An absence of correlation implied that there was no relationship between relative size of the project and magnitude of schedule overruns.

4.10 ANSWER TO RESEARCH QUESTION 6

What are the most significant consequences of the schedule overruns?

The mean of the respondent rating was calculated for each of the consequences of the schedule overruns (question 3.2). The consequence of schedule overruns with the highest mean rating was deemed to be the most significant impact of schedule overruns, the consequence with the second highest mean rating was the second most important, and so on.

The non-parametric Friedman two-way Anova of K-related samples test of equality of medians was performed on the ratings of the consequences of schedule overruns. Further, Wilcoxon matched-pairs signed ranks test was done to confirm the most important consequence(s) of schedule overruns.

4.11 RESPONSE TO QUESTION 3.4

If you had to do this project again, what would you do differently to reduce or eliminate the schedule overruns?

Responses to this question were analyzed and categorized.

CHAPTER 5

RESULTS

The results of the study are presented in ten sections. Section 5.1 describes the characteristics of the projects in the sample. The research questions are answered in sections 5.2 to 5.7. Section 5.8 summarizes the responses to question 3.4 in the questionnaire. Section 5.9 presents other relationships of interest found in the data. Section 5.10 presents the main findings of this study.

5.1 PROFILE OF PROJECTS IN THE SAMPLE

The sample in this study consisted of 41 software development projects from thirty two companies in the Ottawa-Carleton area. There was only one response per project and each respondent provided data on one project only. Table 9 shows the frequency distribution of the number of responses received from each organization. The sample was made up of one project from each of 26 companies, two projects from each of four companies, three and four projects from one company each. Figures 3 and 4 show the breakdown of the organizations in the sample according to their lines of business and number of

Table 9

**Frequency Distribution of the Number of Responses Received from Each
Software Development Organization**

Number of responses from a single organization	Number of organizations	Number of projects
1	26	26
2	4	8
3	1	3
4	1	4
Totals	32	41

Figure 3

Main Lines of Business of the Software Development Organizations that had Projects Included in the Sample

Note: The category 'Other' includes software development organizations whose main line of business was one of the following: artificial intelligence, entertainment, financial point of sale, and military and aero-space. The number within parentheses indicate the number of organizations in a particular line of business. Percentages were calculated on a total of 41.

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

Figure 4

**Number of Employees in the Software Development Organizations that had
Projects Included in the Sample**

employees respectively. Data shown in Figure 4 were collected from the OCEDCO directory that was used to contact the respondents.

Of the forty one projects that formed the total sample, 14 projects (34.1%) were custom software projects, six projects (14.6%) were embedded software projects, 13 projects (31.7%) were software developed for a target market and eight projects (19.5%) were other projects.

The most popular software language used in the projects was C (21 projects). Other languages included C++ (7 projects), Visual Basic (5 projects), Pascal (4 projects), Assembly (3 projects), and SQL (3 projects) COBOL, Foxpro, Oracle and proprietary languages. Twenty seven of the projects were coded in only one language, 12 in two and 2 were coded in three languages.

Figure 5 shows the frequency distribution of the size of the projects in the sample relative to the size of projects normally undertaken by the company.

Table 10 provides a summary of the characteristics of the data in the sample. The sample consisted of projects that varied in complexity from 375 SLOC, 1 module and 8 SLOC per module to 5,000,000 SLOC, 4,792 modules and 3,000,000 SLOC per module. The medians for SLOC, number of modules and the average number of SLOC per module were 60,000, 40 and 650 respectively. The average SLOC, number of modules and the average number

Figure 5

Frequency Distribution of the Size of Projects Included in the Sample Relative to the Size of Projects Normally Undertaken by the Company

Table 10**Summary of the Characteristics of the Data in the Sample**

Characteristics	Minimum	Maximum	Median	Average	Standard deviation
Schedule Overruns (weeks)	3	60	16	18.51	13.77
Schedule Overruns as a Percentage of Estimated Duration	5.08%	337.5%	44.4%	65.22%	67.56%
Schedule Overruns as a Percentage of Actual Duration	4.84%	77.14%	30.77%	33.34%	16.73%
Estimated Duration of Project (weeks)	3	130	35	38.54	27.27
Actual Duration of Projects (weeks)	10	164	52	57.05	36.81
Number of SLOC	375	5,000,000	60,000	410,653	948,076
Number of Modules	1	4,792	40	343.95	845.61
Average number of SLOC per Module	8	3,000,000	650	121,000	529,117

of SLOC per module developed in the projects of the sample were 410,653, 343.95 and 121,000 respectively.

Schedule overruns ranged from 3 weeks to 60 weeks. The median of schedule overruns was 16 weeks and the average was 18.51 weeks. As a percentage of estimated time, schedule overruns ranged from 5.08% to 337.5% with a median of 44.4% and an average of 65.22%. The minimum estimated duration was 3 weeks, the maximum was 130 weeks. The median of the estimated duration was 35 weeks and the average was 38.54 weeks. The actual duration ranged from a minimum of 10 weeks to 164 weeks with a median and average of 52 weeks and 57.05 weeks respectively. Figure 6 shows the estimated and actual durations as well as the magnitude of schedule overruns of the projects in the sample (in ascending order of estimated duration).

5.2 RESEARCH QUESTION 1

Table 11 and Figure 7 show the respondent ratings of the reasons for schedule overruns. It is apparent from Table 11 and Figure 7 that the three reasons with the highest mean ratings are: (1) incorrect estimation of development time, (2) changes in specifications, and (3) inability to detect problems early. The mean ratings for these three reasons were 3.78, 3.41 and 3.00 respectively. The next highest mean rating was 2.22 (lack of experience of the team). The non-parametric Friedman two-way Anova of K-related samples

Figure 6

Estimated and Actual Durations and Magnitude of Schedule Overruns of the Projects in the Sample

Table 11

Respondent Rating of the Reasons for Schedule Overruns

No.	Reason for schedule overruns	Respondent rating			Number of respondents (out of 41) who indicated the reason was important. Percentages are given in parentheses
		Mean	Standard deviation	Rank of mean	
1	Changes in specifications	3.41	1.55	2	39 (95.1%)
2	Lack of senior management support	1.29	1.38	11	27 (65.9%)
3	Inadequate resource allocation	2.10	1.59	5	33 (80.5%)
4	Poor project management	2.07	1.51	6	33 (80.5%)
5	Incorrect estimation of development time	3.78	1.27	1	40 (97.6%)
6	Problems with organizing project teams	1.51	1.57	9	28 (69.3%)
7	Inability to detect problems early	3.00	1.61	3	39 (95.1%)
8	Lack of inter-departmental cooperation	1.44	1.57	10	24 (58.5%)
9	Inflated user expectation	1.76	1.62	8	29 (70.7%)
10	Inadequate techniques	1.95	1.45	7	35 (85.4%)
11	Lack of experience of the team	2.22	1.37	4	37 (90.2%)

Figure 7

Respondent Ratings of the Reasons for Schedule Overruns

confirmed that the rating of the 11 reasons were significantly different from each other. Further, Wilcoxon matched-pairs signed ranks test done on reasons ranked 1 and 4, 2 and 4, and 3 and 4 confirmed that the rating of reasons ranked 1, 2 and 3 were significantly different from the rating of reason ranked 4. Hence the three reasons with the highest mean ratings were considered to be the most important reasons for schedule overruns.

Forty of the 41 respondents (97.6%) felt that incorrect estimation of development time caused schedule overruns. Thirty nine of the respondents (95.1%) felt that changes in specifications caused schedule overruns. The inability to detect problems early was reported as a reason for schedule overruns also by 39 of the 41 respondents (95.1%).

Ten respondents mentioned reasons for schedule overruns in addition to those in the questionnaire. These additional reasons were: lack of internal design, intra-departmental organization, not understanding customer needs, very poor technical design of base package, poor vendor support, poor vendor quality, differences in work breakdown structure, attempt to build the perfect system, change in business direction, technical complexity and lack of ability in technical design. The above-mentioned reasons were mentioned only once by the respondents. Poor communication was mentioned twice by respondents.

5.3 RESEARCH QUESTION 2

Table 12 shows the relationship between the importance of each reason in causing schedule overruns and software complexity.

The most important findings are:

1. There is no relationship at $\alpha = 0.05$, between the three most important reasons for schedule overruns (incorrect estimation of development time, changes in specifications and inability to detect problems early) and the three measures of complexity.
2. Nine of the eleven reasons for schedule overruns were not correlated with any of the three proxies of software complexity at $\alpha = 0.05$. These nine reasons were: changes in specifications, poor project management, incorrect estimation of development time, problems with organizing project teams, inability to detect problems early, lack of inter-departmental cooperation, inflated user expectation, inadequate techniques and lack of experience of the team.
3. There is a positive correlation at $\alpha = 0.05$ between lack of senior management support and two proxies for software complexity. As the number of SLOC and average number of SLOC per module increased, the importance of lack of senior management support as a cause of schedule overruns increased.

Table 12
Spearman Rank Order Correlation Between the Importance of Each Reason for
Schedule Overruns and Complexity

No.	Reason for schedule overruns	Measures of Complexity		
		Number of SLOC	Number of modules	Average number of SLOC per module
1	Changes in specifications	-.0135 (.933)	-.0571 (.723)	-.0426 (.792)
2	Lack of senior management support	.3753 (.016)	-.0895 (.578)	.3255 (.038)
3	Inadequate resource allocation	.2392 (.132)	-.0725 (.653)	.2727 (.085)
4	Poor project management	.1725 (.281)	-.0682 (.672)	.0209 (.897)
5	Incorrect estimation of development time	.0796 (.621)	-.0910 (.572)	.1863 (.243)
6	Problems with organizing project teams	.1865 (.243)	-.0791 (.623)	.2504 (.114)
7	Inability to detect problems early	.0155 (.923)	.1315 (.412)	-.1646 (.304)
8	Lack of inter-departmental cooperation	-.0558 (.729)	-.1636 (.307)	.0217 (.893)
9	Inflated user expectation	.1021 (.525)	-.1760 (.271)	.1160 (.470)
10	Inadequate techniques	-.0011 (.994)	.0177 (.913)	-.0375 (.816)
11	Lack of experience of the team	-.0870 (.589)	-.0145 (.928)	-.0812 (.614)

Significance levels are given within parentheses. n = 41.

4. The average number of SLOC per module was positively correlated with inadequate resource allocation at $\alpha = 0.085$. As the average number of SLOC per module increased, the importance of inadequate resource allocation, as a reason for schedule overruns, increased.

5.4 RESEARCH QUESTION 3

Table 13 shows the correlation between the absolute magnitude of schedule overruns, magnitude of schedule overruns expressed as a percentage of estimated and actual durations and software complexity. There was a positive correlation at $\alpha = 0.05$, between the average number of SLOC per module and magnitude of schedule overruns expressed as a percentage of actual duration. There was a positive correlation, at $\alpha = 0.10$, between the average number of SLOC per module and magnitude of schedule overruns expressed as a percentage of estimated duration. As the average number of SLOC per module increased, magnitude of schedule overruns as a percentage of estimated and actual durations also increased.

There was no relationship between the absolute magnitude of schedule overruns and the three software complexity measures.

Table 13

Pearson Correlation Between the Magnitude of Schedule Overruns, Magnitude of Schedule Overruns Expressed as a Percentage of Estimated and Actual Durations and Complexity

	Measures of complexity		
	SLOC	Number of modules	Average number of SLOC per module
Magnitude of schedule overruns	.0968 (.547)	.1951 (.222)	-.0506 (.754)
Magnitude of schedule overruns expressed as a percentage of estimated duration	.0935 (.561)	-.0588 (.715)	.2807 (.075)
Magnitude of schedule overruns expressed as a percentage of actual duration	.1375 (.391)	-.0441 (.784)	.3100 (.049)

Significance levels are given in parentheses. n = 41.

5.5 RESEARCH QUESTION 4

Table 14 shows the relationship between the importance of a reason for schedule overruns and the magnitude of schedule overruns. There was a positive correlation between the importance of problems with organizing project teams and the magnitude of schedule overruns at $\alpha = 0.05$. As the importance of problems with organizing project teams as a reason for schedule overruns increased, the magnitude of schedule overruns also increased.

There was a negative correlation ($\alpha = 0.06$) between the importance of lack of experience of the team and the magnitude of schedule overruns. As the magnitude of schedule overruns increased, the importance of lack of experience of the team as a reason for schedule overruns decreased.

Table 14**Spearman Rank Order Correlation Between the Importance of a Reason for Schedule Overruns and the Magnitude of Schedule Overruns**

	Reasons for schedule overruns	Magnitude of schedule overruns
1	Changes in specifications	-.0744 (.644)
2	Lack of senior management support	.0587 (.716)
3	Inadequate resource allocation	.0749 (.642)
4	Poor project management	.0665 (.680)
5	Incorrect estimation of development time	-.0814 (.613)
6	Problems with organizing project teams	.3222 (.040)
7	Inability to detect problems early	.1904 (.233)
8	Lack of inter-departmental cooperation	.1457 (.363)
9	Inflated user expectation	-.0661 (.682)
10	Inadequate techniques	.0259 (.872)
11	Lack of experience of the team	-.3061 (.052)

Significance levels are given within parentheses.

5.6 RESEARCH QUESTION 5

Table 15 shows that the size of the projects in the sample relative to the size of other projects normally undertaken by the company and the magnitude of schedule overruns were not correlated.

Table 15

Spearman Rank Order Correlation Between the Magnitude of Schedule Overruns and Size of Project Relative to the Size of Projects Normally Undertaken by the Company

	Relative size of project
Magnitude of schedule overruns	.1305 (.416)

The significance level is given within parenthesis.

5.7 RESEARCH QUESTION 6

Table 16 provides the mean respondent rating of the consequences of schedule overruns. The most significant consequence of the schedule overruns was on project cost (mean of respondent rating = 3.63). Customer satisfaction (mean of respondent rating = 3.12), and financial success (mean of respondent rating = 3.03) progress of dependent projects (mean respondent rating = 2.71) were ranked second, third and fourth respectively. Two of the 41 respondents did not provide a rating for the significance of financial success as a consequence of schedule overruns. The mean respondent rating for the significance of financial success was calculated on 39 responses while the mean respondent ratings for the other consequences were calculated on 41 responses.

The non-parametric Friedman two-way Anova of K-related samples test confirmed that the differences between the means was significant at $\alpha = 0.08$ ($n = 39$). Wilcoxon matched-pairs signed ranks test confirmed that the median of project cost was significantly different from that of the other three consequences ($\alpha = 0.06$). There was no significant difference between the medians of the other three consequences.

Some of the other consequences mentioned by the respondents are: loss of morale of the members on the development team and loss of credibility

Table 16

Respondent Rating of the Consequences of Schedule Overruns

No	Consequence of schedule overruns	Respondent rating		
		Mean	Standard deviation	Rank of mean
1	Project cost	3.63	1.39	1
2	Customer satisfaction	3.12	1.33	2
3	Progress of dependent projects	2.71	1.52	4
4	Financial success	3.03	1.44	3

n = 41 for project cost, customer satisfaction and progress of dependent projects.

n = 39 for financial success.

(5), inability to allocate organizational resources optimally (2), inability to reach markets on time, loss of potential business (2) and quality (2). The number within parentheses indicates the number of respondents who mentioned it.

Two respondents reported positive outcomes. Schedule overruns resulted in a problem-free product and the opportunity to make changes to requirements.

5.8 RESPONSE TO QUESTION 3.4

Question 3.4 was "If you had to do this project again, what would you do differently to reduce or eliminate the schedule overruns?" The responses to this question were organized into eight categories. Six categories were initially defined by the reasons for schedule overruns used in the questionnaire. Two other categories were added to organize responses that did not fit into the six categories. The categories added were: product characteristics and development process.

5.8.1 Changes in specifications

1. Do not expect the client or customer to provide complete and correct requirements.
2. Ensure that even detailed customer requirements are thoroughly analyzed and the problem domain is well understood. Review specifications as thoroughly as possible to detect errors early rather than have them surface

once development is underway. Ensure that the product is well defined, that requirements are frozen and that the scope of the project is reasonable before development commences.

3. Whenever changes to specifications are required, ensure that these changes are controlled.

5.8.2 Problems with organizing project teams, inadequate resource allocation and lack of experience of the team

1. Wherever possible use experienced resources. If this cannot be done, it is expeditious to train staff on new development tools before the actual development begins rather than train them on the job (during the development project).
2. Have a team whose members are dedicated to the development project full-time. If this is not possible establish a process for managing both multi-project work as well as non-project activities such as maintenance and support.
3. Ensure that adequate personnel resources are allocated when the project plans require them. Adding more people later to make up for lost time increases the schedule overruns even further.
4. Make project team members accountable for schedule overruns.

5.8.3 Incorrect estimation of development time

- 1. Improve techniques by which schedule estimates are made. Keep track of schedule estimates, deviations from schedule estimates, reasons for deviations. Review these data for past projects to learn and improve schedule estimation procedures.**
- 2. Build prototypes before committing to a schedule.**
- 3. Have people who are not part of the development team also estimate the development time for the project.**
- 4. Review estimates with senior management, especially when unfamiliar tools are used for development. Increase the risk factor in estimates when new development technology is involved.**
- 5. Ensure better time management.**
- 6. Make generous time allowances to accommodate unforeseen problems.**

5.8.4 Lack of Inter-departmental cooperation

- 1. Ensure that interdependencies amongst the members of the development team and between the development team and clients, other groups or individuals, critical for project completion, are clearly understood.**
- 2. Clearly define roles and responsibilities between design groups.**
- 3. Clearly communicate priority of work activities to different design groups.**

4. Ensure that the groups or vendors that the development team depends on are reliable and deliver as promised.
5. Avoid direct communication between the programmer and the client.

5.8.5 Inability to detect problems early

1. Develop more efficient test cases. Improve system testing techniques.
2. Include an extensive testing stage with all possible configurations (hardware and software).

5.8.6 Poor project management

1. When the project is not defined clearly, the development organization should protect itself by entering into a cost-plus contract rather than a fixed-price contract. Make provisions for revising development schedules.
2. Have weekly or even daily project reviews so that any deviation from the plan is immediately detected and corrected. Monitor project milestones closely. Report problems as soon as they arise.
3. Break large projects into smaller ones.
4. If the development group is large, an experienced manager should lead the team.

5.8.7 Product characteristics

- 1. Reduce complexity of the development as much as possible, by reducing user or client expectations.**
- 2. Arrange to deliver the product in stages. The first release incorporates the most important functions and functionality may be improved in the subsequent releases.**

5.8.8 Development process

- 1. Allocate more time for the investigation of design issues, for high level design and in its validation. Have experienced designers review and validate design. Maintain continuous contact with the customer during the high level design process. Do not accept changes once high level design is frozen.**
- 2. Ensure that any off-the-shelf package is thoroughly evaluated by the staff of the development organization for reliability of performance and completeness of documentation before any time schedules for customization are provided. Never assume that the vendor's product has been fully tested and allow for uncertainties in vendor product quality in spite of vendors' assurances. The same applies when using a new platform for software development.**
- 3. Ensure that the design process is related to the strengths and weaknesses of the software development tools.**
- 4. Make more use of Object-Oriented programming languages and tools.**

5.9 OTHER RELATIONSHIPS OBSERVED

This section presents some interesting findings shown in Appendices E, F and G. These findings were by-products of the analysis used to answer the research questions.

Findings from Appendix E

Appendix E shows the intercorrelations among the importance of reasons for schedule overruns.

The importance of lack of senior management support is positively correlated with the importance of inadequate resource allocation, poor project management, lack of inter-departmental cooperation, inflated user expectation and lack of experience of the team ($\alpha = 0.05$). This result indicates the importance of senior management support in assuring adequate resource allocation and sound project management, in ensuring that different departments cooperate with the development team, in containing user expectation and in assigning experienced personnel to the development team.

The importance of poor project management is positively correlated with the importance of lack senior management support, inadequate resource allocation, inability to detect problems early, inflated user expectation and inadequate techniques at $\alpha = 0.05$. This result indicates that good project management has an important role to play in detecting problems early,

containing user expectation and ensuring that adequate techniques are available to the development team.

The importance of inadequate resource allocation was positively correlated to lack of senior management support, poor project management, incorrect estimation of development time and problems with organizing project teams $\alpha = 0.05$. This result suggests that inadequate resource allocation and incorrect estimation of development time are associated. Further, poor project management and the problems with organizing project teams are also associated with inadequate resource allocation.

The inability to detect problems early was positively correlated to incorrect estimation of development time at $\alpha = 0.05$.

Findings from Appendix F

Appendix F shows the relationship between the importance of a reason for schedule overruns and the size of the project relative to the size of other projects normally undertaken by the company.

A significant positive correlation was found between the importance of inadequate resource allocation as a reason for schedule overruns and relative size of project $\alpha = 0.062$. As the relative size of the project increased, the importance of inadequate resource allocation as a reason for schedule overruns also increased.

Findings from Appendix G

Appendix G shows the relationships between the consequences of schedule overruns and the importance of a reason for schedule overruns.

A positive correlation was found between the significance of project cost and the importance of poor project management and the importance of incorrect estimation of development time ($\alpha = 0.05$). As the significance of project cost as a consequence of schedule overruns increased, the importance of poor project management and that of incorrect estimation of development time as reasons for schedule overruns also increased. This result suggests that increased project cost is associated with poor project management and incorrect estimation of development time.

A positive correlation was found between the significance of customer satisfaction and the importance of inadequate resource allocation ($\alpha = 0.05$). As the significance of customer satisfaction as a consequence of schedule overruns increased, the importance of inadequate resource allocation as a reason for schedule overruns increased. This result indicates that customer dissatisfaction is associated with inadequate resource allocation.

A positive correlation was found between the significance of financial success and the importance of lack of senior management support, poor project management and lack of experience of the team ($\alpha = 0.05$). As the significance

of financial success as a consequence of schedule overruns increased, the importance of lack of senior management support, poor project management and lack of experience of the team also increased. This result suggests that financial success is associated with senior management support, good project management and a well experienced team.

There was a significant positive correlation between the significance of progress of dependent projects and the importance of incorrect estimation of development time. Whenever, the importance of incorrect estimation of development time as a reason for schedule overruns increased, the progress of dependent projects was affected.

5.10 MAIN FINDINGS

This section is divided into two parts. The first part lists the main findings that pertain to the research questions in Figure 1. The second part lists the findings that pertain to appendices E, F and G.

The main findings that pertain to the research questions

1. The three most important reasons for schedule overruns in software development projects were incorrect estimation of development time, changes in specification and inability to detect problems early. There was no relationship between these three reasons for schedule overruns and the three proxies for complexity at $\alpha = 0.05$.

- 2. The importance of lack of senior management support as a reason for schedule overruns increased as the number of SLOC and the average number of SLOC per module increased ($\alpha = 0.05$).**
- 3. The importance of inadequate resource allocation as a reason for schedule overruns increased as the average number of SLOC per module increased ($\alpha = 0.085$).**
- 4. The magnitude of schedule overruns expressed as a percentage of actual and estimated duration was positively correlated with the average number of SLOC per module at $\alpha = 0.05$ and $\alpha = 0.10$ respectively. There was no association between the absolute magnitude of schedule overruns and the three measures of complexity.**
- 5. As the magnitude of schedule overruns increased, the importance of problems with organizing project teams as a cause of schedule overruns increased ($\alpha = 0.05$).**
- 6. As the magnitude of schedule overruns increased, the importance of lack of experience of the team as a cause of schedule overruns decreased ($\alpha = 0.052$).**
- 7. There is no relationship between the size of the project, relative to the size of projects normally undertaken by the company, and the magnitude of schedule overruns.**

8. Schedule overruns affected project costs the most. There was no significant difference between the medians of the other three consequences: customer satisfaction, financial success and the progress of dependent projects.
9. From the responses to question 3.4 it is apparent that the development process characteristics such as quality and thoroughness of the execution of the development process and product characteristics affect adherence to schedule.

The main findings that pertain to Appendices E, F and G

10. The importance of lack of senior management support was positively correlated to inadequate resource allocation, poor project management, lack of inter-departmental cooperation, inflated user expectation and lack of experience of the team ($\alpha = 0.05$).
11. The importance of poor project management was positively correlated to lack of senior management support, inadequate resource allocation, inability to detect problems early, inflated user expectation and inadequate techniques ($\alpha = 0.05$).
12. The importance of inadequate resource allocation was positively correlated to lack of senior management support, poor project management, incorrect estimation of development time and problems with organizing project teams ($\alpha = 0.05$).

- 13. The inability to detect problems early was positively correlated to incorrect estimation of development time at $\alpha = 0.05$.**
- 14. As the relative size of the project increased, the importance of inadequate resource allocation as a reason for schedule overruns also increased ($\alpha = 0.062$).**
- 15. As the significance of project cost as a consequence of schedule overruns increased, the importance of poor project management and incorrect estimation of development time as reasons for schedule overruns also increased ($\alpha = 0.05$).**
- 16. As the significance of customer satisfaction increased, the importance of inadequate resource allocation as a reason for schedule overruns also increased ($\alpha = 0.05$).**
- 17. As the significance of financial success as a consequence of schedule overruns increased, the importance of lack of senior management support, poor project management and lack of experience of the team as reasons for schedule overruns also increased ($\alpha = 0.05$).**
- 18. As the significance of progress of dependent projects as a consequence of schedule overruns increased, the importance of incorrect estimation of development time as a reason for schedule overruns also increased ($\alpha = 0.05$).**

CHAPTER 6

DISCUSSION OF RESULTS

This chapter discusses the results of this study. Section 6.1 presents a summary of the results of this research. Sections 6.2 to 6.8 discuss the salient results of the study.

6.1 SUMMARY OF IMPORTANT FINDINGS

The average schedule overrun in this study is 65% of estimated duration. It is higher than the 17% and 50% reported by Paulish and Carleton (1994) and Gibbs (1994) respectively. However, it is well below the 100% to 200% reported by Simmons, Ellis and Escamilla (1993). The range of schedule overruns in this study is 5% to 337% of estimated duration. It is wider than the 13% to 42% reported by Paulish and Carleton (1994). The wide range of schedule overruns may be attributed to the fact that this research included only projects that experienced schedule overruns.

Figures 8 and 9, and Table 17 summarize the results obtained in this study. Figure 8 provides the answers to the six research questions that were studied in this research. The + and - signs indicate the direction of correlation

Figure 8

Summary of Answers to the Research Questions

Note: Levels of significance are given within parentheses. Arrows indicate causality assumed in this research. Lines indicate associations.

* Represents the level of significance of the correlation between average number of SLOC per module and the magnitude of schedule overruns expressed as a percentage of actual duration.

** Represents the level of significance of the correlation between average number of SLOC per module and the magnitude of schedule overruns expressed as a percentage of estimated duration

between the variables in the boxes. The numbers in parentheses are significance levels. The arrows indicate causality assumed in this study. The lines indicate associations.

Figure 9 presents the main findings that pertain to Appendices E, F and G. The circles represent the reasons for schedule overruns. The rectangles represent the consequences of schedule overruns. The hexagon represents a project characteristic (size of project relative to the size of projects normally undertaken by the company). The lines between the circles, rectangles and the hexagon represent positive correlations significant at $\alpha = 0.05$ with one exception. The exception is the correlation between the 'relative size of a project' and 'inadequate resource allocation' which is significant at $\alpha = 0.062$.

Figure 9

Summary of Findings that Pertain to Appendices E, F and G

- Reasons for schedule overruns
- ▭ Consequences of schedule overruns
- ⬡ Project characteristic (relative size)

Note: All correlations are positive and significant at $\alpha = 0.05$ except the correlation between inadequate resource allocation and relative size of the project which is significant at $\alpha = 0.062$.

Table 17 represents a summary of the suggestions made by the respondents to reduce or eliminate schedule overruns if they had to re-do the project for which they had provided data (response to question 3.4).

Two-thirds of the twenty-nine suggestions listed in Table 17 relate to non-technical project management issues. The suggestions that are non-technical in nature have been marked with an 'N'. The other seven suggestions that are technical in nature have been marked with a 'T'. It is apparent from Table 17 that non-technical project management issues dominate the concerns of software development project managers as far as overcoming schedule overruns is concerned.

Among the suggestions presented in Table 17 are a few that have not been addressed in the literature that was reviewed for this thesis. Some of the suggestions in Table 17 under 'Product Characteristics' and 'Development Process' were different from those identified in the literature suggesting that there may be other important reasons for schedule overruns that have not been previously studied. The most salient examples are poor assessment of vendor tools and documentation, lack of validation of high level design and inability to arrange the delivery of the product in stages.

Table 17
Summary of Responses to Question 3.4

	CHANGES IN SPECIFICATIONS
N	Expect customer requirements to be incomplete
T	Review customer requirements thoroughly and clearly define the product
N	Understand the problem domain well
N	Agree on procedure to incorporate changes in specifications after development begins
	PROBLEMS WITH ORGANIZING PROJECT TEAMS, INADEQUATE RESOURCE ALLOCATION AND LACK OF EXPERIENCE OF THE TEAM
N	Train personnel before actual development begins - not on-the-job
N	Project team members should be dedicated to the project full-time
N	Allocate resources when required by the project plan
N	Make project personnel accountable for schedule overruns
	INCORRECT ESTIMATION OF DEVELOPMENT TIME
N	Review data from past projects to learn and improve schedule estimation procedures
T	Build prototypes before committing to a schedule
N	Have designers who are not part of the development team review schedule estimates
N	Review schedule estimates with senior management or increase risk factor when new development technology is involved
N	Provide slack to accommodate unforeseen problems
	LACK OF INTERDEPARTMENTAL COOPERATION
N	Ensure that interdependencies are clearly understood
N	Clearly define roles and responsibilities between design groups
N	Communicate clearly the work priorities between the design groups
N	Ensure that all parties that the development group depends on are reliable
N	Avoid direct communication between the programmer and the client
	INABILITY TO DETECT PROBLEMS EARLY
T	Develop efficient test cases
T	Include an extensive testing stage with both software and hardware configurations
	POOR PROJECT MANAGEMENT
N	Enter into fixed-price contracts when the project is not defined clearly
N	Have weekly or daily milestone reviews to detect problems early
N	Break large projects into smaller ones
N	Experienced manager should lead large development teams
	PRODUCT CHARACTERISTICS
N	Reduce client expectations to control complexity
N	Deliver product in stages
	DEVELOPMENT PROCESS
T	Validate high level design
T	Assess vendor tool and off-the-shelf-product capability independent of documentation
N	Ensure that the design process is related to the strengths and weaknesses of the software development tools
T	Use Object-Oriented methods and tools

Note: 'N' Represents suggestions that are non-technical in nature
'T' Represents suggestions that are technical in nature

6.2. SCHEDULE ESTIMATION PROCEDURES

The results of this research emphasize the need to develop rigorous schedule estimation procedures for software development projects. Incorrect estimation of development time was rated the most important reason for schedule overruns. Table 12 shows that the importance of incorrect estimation of development time as a reason for schedule overruns did not vary with the complexity of the project.

The finding that incorrect estimation of development time is an important reason for schedule overruns supports the findings reported in earlier studies. Over-optimistic planning was reported as an important reason for schedule overruns by Phan, Vogel and Nunamaker (1988). In the study by Thamhain and Wilemon (1986), unrealistic project plan was ranked as the second and third most important reason for schedule overruns by general managers and product managers respectively.

Incorrect estimation of development time may be attributed to erroneous assumptions made by managers, lack of understanding of the problem domain and complexity.

Erroneous assumptions

In 1975, Brooks Jr. was the first to point out that procedures used in estimating the duration of software development were poorly developed. The

three erroneous assumptions that managers make while estimating development schedule are: 1. person and month, in the person-months estimates of the effort required to develop the software product, are interchangeable, 2. the entire work week of the programmer is presumably spent in developing, testing and debugging and 3. the schedule estimates for small projects can be extrapolated to derive schedule estimates for larger projects.

According to Brooks Jr. (1975), managers often make schedule estimates by dividing the effort required (in person months) by the number of persons available. When they do so, managers make the assumption that if it takes one person 5 months to complete a task the same task could be completed by 5 persons in one month. This assumption overlooks the sequential interdependencies between tasks. It also presupposes that the task can be divided into five completely independent modules with little need for communication between the people developing them. Since design partitioning into completely independent modules is seldom achieved, the need for communication between developers cannot be eliminated. The greater the number of people on a project the more the time that is spent on non-programming tasks such as communication. Less time is spent on actual development. Frequently, the result of increasing the number of persons working on a project is that the project is not completed as estimated.

According to a study by Charles Portman, quoted by Brooks Jr. (1975), only a portion of a programmer's work week is spent on actual development. A considerable amount of the work week is taken up by machine down time, unrelated work such as maintenance, meetings, paperwork, company business, sickness, personal time, and so on. Since this fact is often overlooked when schedule estimates are made, the actual duration is longer than the estimated duration.

Another erroneous assumption frequently made by managers is that schedule estimates of small projects can be extrapolated to larger projects (Sackman, Erikson and Grant 1968). If a product requiring a hundred lines of code took one person 10 days to develop, test and debug, the assumption is that a product requiring a thousand lines of code can be completed in a hundred days. In extrapolating the development duration of the smaller project, the increased amount of time required for planning, documentation, testing, system integration and training is overlooked.

Lack of understanding of the problem domain

Boehm (1984) attributes incorrect cost estimation to lack of understanding of the software job to be done. While he refers to cost estimates, the same is also true for schedule estimates. We surmise that, the variability of schedule

estimates made early in the feasibility stage is high due to insufficient knowledge about the software job to be done.

There is a need for a knowledge acquisition stage early in the development process. Knowledge about the problem domain improves the designers' and managers' understanding of the job to be done. Discussions with people who are familiar with managing software development processes reveal that the methods of improving the teams' understanding of the problem domain include prototyping and input from domain experts.

Complexity

This study found that the importance of incorrect estimation of development time is invariant across projects that differ in complexity. This finding has not been reported by research studies undertaken previously. It appears that incorrect estimation of development time is an important reason even in less complex projects. This reaffirms the need to improve estimation procedures for software development projects .

6.3 MANAGEMENT OF THE CHANGES IN SPECIFICATIONS

The results of this research suggest that there is a need to better manage the changes in specifications.

Table 11 shows that changes in specifications was the reason with the second highest mean rating (3.41), and 95.1% of the respondents felt that it was

a cause of schedule overruns. This result is consistent with those reported by Gupta and Wilemon (1990), Thamhain and Wilemon (1986) and Phan, Vogel and Nunamaker (1988). The finding shown in Table 12 that the importance of changes in specifications as a reason for schedule overruns did not vary with the complexity of the project has not been reported by previous studies.

Changes in specifications are a part of software development for two reasons. First, customer requirements are incorrect and inadequate because customers are unable to articulate the requirements fully and correctly in a manner that is understood by designers. Second, customer requirements are misinterpreted or omitted during initial requirements analysis because few designers have sufficient expertise in the problem domain to ask customers the right questions to elicit and refine the requirements. As the customer and the development team interact with prototypes or portions of the software, new requirements are discovered, and existing requirements are either modified or changed. Therefore specifications evolve during the development process.

The challenge is to reduce inaccurate customer requirements, to accommodate evolving specifications in a controlled manner and to protect against contractual penalties resulting from changes in specifications.

The quality of the analysis of customer requirements must improve. Frequent changes in specifications due to late discovery of customer

requirements should be reduced. This may be done by seeking input from domain experts, locating team members at customer organizations and encouraging direct contact between the designers and key customers (Bailetti and Guild 1991).

Building prototypes before full scale development begins offers many advantages. Prototyping gives the development team an idea of the effort required for developing the product (Boehm 1984) and provides a tool to elicit and refine customer requirements (Rowen 1990, Bennatan 1992). Carmel and Becker (1995) suggest involving customers and the marketing department early in the development process. This involvement should continue until the specifications are finalized and signed-off. This allows the development team to discuss with customers and other stake holders the effect of implementing the changes in specifications on schedules.

Full scale development should not begin until specifications are frozen. Fixed price contracts should stipulate the stage in the development process beyond which changes in specifications would not be accommodated in the current release of the software. It should also stipulate the process by which changes in specifications will be accommodated. The effect of implementing the changes on development schedule should be thoroughly evaluated and all stake holders should be informed of these changes as early as possible. This will

enable customers and managers of other software projects to take appropriate action to mitigate the effects of late delivery.

6.4 FRONT-END DEVELOPMENT PROCESS

There is a need to detect problems early in the development process, especially in projects with a large number of modules. The inability to detect problems early was the reason for schedule overruns with the third highest mean rating (Table 11). In addition, 95.1% of the respondents felt that it was a cause of schedule overruns. Inability to detect problems early was ranked fifth by project managers in the study by Thamhain and Wilemon (1986).

Table 12 shows that the importance of inability to detect problems early was invariant across projects that differed in complexity. This finding does not support results of earlier research. Korson and Vaishnavi (1986) found that the time required to modify modular versions of a software was less than that required to modify monolithic versions of the software. The results of Card, Page and McGarry (1985) were just the opposite of that reported by Korson and Vaishnavi (1986).

Appendix E shows that as the importance of inability to detect problems early as a reason for schedule overruns increased, the importance of poor project management also increased.

Inability to detect problems early and complexity

The findings of this research emphasize the need to devote more time and effort to detect problems early irrespective of the complexity of the project. It appears that late problem detection of errors is an important reason for schedule overruns even in seemingly less complex projects.

Early detection of programming errors

Programming errors are of two types: syntactic and semantic. Syntax errors are usually discovered during compilation and require little effort to correct. Semantic errors (computational, logical and data) are more difficult to discover and rectify. Encouraging developers to self-test and correct their code enables early error detection and correction. Self-testing increases the programmers' awareness of errors and may also help reduce the number of errors.

Among the problems encountered by software development projects are:

1. insufficient validation of high level design
2. difficulty in developing test cases
3. performance of vendor tools below expectation and below levels stated in the documentation

Validation of high level design

Many of the errors that are discovered late can be traced back to errors made during the high level design stage. A thorough validation of high level design ensures that such errors are discovered early in the development process when the time and effort required to correct them are considerably less. Experienced designers should be assigned the task of validating the high level design.

Developing test cases

Test cases are usually developed to detect errors. Test cases that scour every aspect and every part of the program are not easy to develop. Closer interaction with the customer to understand the problem domain will help build better test cases. It will also help test the software under actual conditions of use by creating better simulations.

Evaluating vendor tools

Often development projects rely on vendor tools for building the software or for developing applications. However, vendor tools may not perform at the levels stated in their documentation. Assessing the capability of vendor using such documentation has led to problems later in the development project. Project managers should therefore, ensure that all vendor tools are tested by the

development team to evaluate their capabilities independent of vendor documentation.

6.5 ADDRESSING COMPLEXITY

The results of this research identified the need for capability to represent and understand the increasing complexity of software. Table 13 shows that the magnitude of schedule overruns expressed as a percentage of estimated and actual durations increased as the average number of SLOC per module increased. The more complex projects were associated with higher percentages of schedule overruns. The findings of this study are consistent with those reported by van Genuchten (1991) and Thamhain and Wilemon (1986). They found that there was a tendency to underestimate the complexity and scope of projects.

Table 13 also shows that there was no correlation between the absolute magnitude of schedule overruns and SLOC, number of modules and average number of SLOC per module. This result is similar to that found by Jenkins, Naumann and Wetherbe (1984). Their study found that schedule overruns were uniformly distributed among large, medium and small projects.

Even when adequate information is available, estimating complexity is not an easy task. Smaller and less complex tasks may be grasped as a cognitive gestalt (Curtis, Sheppard and Milliman 1979), but as the complexity increases

comprehension becomes more difficult. Booch (1994) attributes schedule overruns, among other problems, to the inability to master complexity.

There is a need for methods for representing complexity and tools that can process high level abstractions. Complex problems may be broken down into smaller and simpler modules each of which may be refined independently. Object-oriented (O-O) methods and tools exploit the inherent hierarchical nature of complex systems to decompose them into simpler chunks. Standardized notation used to represent complex systems within O-O methods provides a common instrument for designers to communicate with each other to discuss and understand the problem better. An O-O representation of the problem uses terms directly drawn from the problem domain. The use of terms directly drawn from the problem domain reduces the semantic gap between designers and customers.

6.6 ORGANIZATIONAL LEARNING

The results of this study suggest that there is a need to transfer and share the knowledge gained in one project to other projects in the organization. Size of the software development project relative to the size of other software development projects normally undertaken by the company was not related to the schedule overruns (Table 15). This result has not been reported in the literature previously.

Figure 5 shows that half the projects in the sample were either similar or smaller to projects normally undertaken by the company. Two possible explanations for the absence of an association between relative size and magnitude of schedule overruns are:

- 1. each project is so different from others that little of the past learning is transferable or**
- 2. there is little transfer of knowledge from one project to another.**

There is support for the latter explanation - each project is 'individually crafted' (Booch 1994). During informal conversations with some of the respondents in this study, the researcher found that very little project information was being recorded and maintained. Most projects were being managed without the benefit of knowing how the other projects had been managed.

There is a need to collect schedule overrun data to assist project managers in developing norms that can be used to manage projects. Analysis of the reasons why past software development projects overran their schedules and analysis of the magnitude of such overruns can provide managers with information needed to plan and estimate schedules better. Deviations from plan should be corrected immediately to prevent them from going out of control. Expert systems that can unobtrusively monitor and provide valuable project

management information to managers have been available for a long time. However, their use is not that prevalent.

6.7 RESOURCE ALLOCATION

As the importance of incorrect estimation of development time as a reason for schedule overruns increased so did the importance of inadequate resource allocation (Appendix E). This finding contradicts an earlier finding by Jenkins, Naumann and Wetherbe (1984). They found that inadequate resources were a reason for schedule overruns, but inadequate resources were not related to poor estimates. Perhaps the resources for the projects in the Jenkins, Naumann and Wetherbe study (1984) were correctly estimated but the resources indicated by the estimates were not allocated.

Unlike the projects in the Jenkins, Naumann and Wetherbe study (1984), projects in this research sample appear to have been allocated as much resources as were estimated. Since there is evidence that schedule estimates were incorrect, there is reason to believe that estimates of resources were also incorrect. This belief is supported by the fact that inadequate resource allocation was also an important reason for schedule overruns in projects that were either large or very large compared to the projects normally undertaken by the organization (Appendix F).

A study by Sackman, Erikson and Grant (1968) points out that the effort required increases at a faster rate than the rate at which as the number of SLOC increase. I surmise that the resources required increases at a rate faster than the rate at which the size of the project increases.

6.8 PERFORMANCE OF SOFTWARE DEVELOPMENT TEAMS

The results of this study indicate that there is a need to improve the overall performance of software development teams.

Associations between the importance of lack of senior management support and poor project management, and inadequate resource allocation, lack of inter-departmental cooperation, lack of experience of the team, incorrect estimation of development time, problems with organizing project teams and inability to detect problems early (Appendix E) show that the performance of software development teams can be improved.

Similarly, associations between the importance of lack of senior management support, poor project management, inadequate resource allocation, incorrect estimation development time lack of experience of the team, and increased project costs, customer dissatisfaction and financial success (Appendix G) show that the important consequences of schedule overruns can be addressed by improving the performance of software development teams.

CHAPTER 7

CONCLUSION

The objective of this study was to examine the reasons for schedule overruns in software development projects. This chapter is the conclusion to this study. Section 7.1 presents the recommendations to managers, section 7.2 includes the suggestions for further research and 7.3 addresses the limitations of this study.

7.1 RECOMMENDATIONS TO MANAGERS

The recommendations that follow are based on the major findings of this research. The major findings of this research suggest that there is a need to:

- improve schedule estimation procedures
- reduce and control changes in specifications
- improve front-end development processes
- deal with the increasing complexity of software
- transfer the knowledge gained in one project to other development projects in the company
- build high performance software development teams

7.1.1 Maintain schedule management database

Maintain a database that includes information on the following:

- **Information considered in developing schedule estimates**
- **Time spent on acquiring the information used to develop the schedule estimate**
- **Schedule estimate data**
- **Actual schedule data**
- **Reasons that account for the difference between estimated and actual software delivery**
- **Preventive and restorative measures taken to overcome schedule overruns**
- **Evaluation of the effectiveness of the preventive and restorative measures taken**
- **Project characteristics data such as size and task difficulty**
- **Lessons learnt about managing the schedule of this project that can be used to manage other projects in the company.**

The data for this database can be collected by using a computerized management system such as Computerized life cycle Advising, Monitoring and Predicting system (Chen 1985). Such a system can unobtrusively monitor software development projects, collect and analyze the data and advise managers in setting norms for process parameters. The analysis of these data

should focus on the factors that affect schedule estimates. The results of the analysis will serve as the basis for developing norms to guide project control. As more data are gathered and analyzed, there will be sufficient information to reduce schedule overruns. As Boehm (1984) suggests, software engineers should themselves be customers of management information to guide and support the many decisions required in the course of a software development project.

7.1.2 Build prototypes before full scale development starts

Prototypes help the development team to better understand the problem domain, customer requirements (Bennatan 1992) and the effort required to develop the product (Boehm 1984). Once the problem domain and customer requirements are understood the development team can provide more accurate schedule estimates. Prototypes can also help in the evaluation and testing of vendor tools and new operating systems.

7.1.3 Increase the development team's knowledge of the problem domain

Better knowledge of the problem domain reduces the number of errors due to faulty requirements analysis and helps build more effective test cases. When the development team understands the problem domain, it is better

equipped to ask the customers the right questions to elicit and refine requirements and to interpret customer requirements.

By understanding the actual environment of use of the software the testing department can develop better and more comprehensive test cases and simulate the conditions of use of the software.

Methods used to enhance the domain knowledge of software development teams include consulting domain experts, locating team members at customer sites and face-to-face meetings between designers and key customers (Bailetti and Guild 1991).

7.1.4 Inform stake holders early of changes in specifications

Full scale development should not begin until specifications have been frozen. Implementation of changes in specifications during development should be thoroughly evaluated. All the stake holders should be informed of the effect of these changes on project schedule as early as possible. This enables external customers and managers of other software projects to take appropriate action to mitigate the effects of late delivery.

Development contracts should stipulate clearly how far down the development process changes to specifications will be accommodated. the contract should also specify the process by which changes in specifications would be implemented.

7.1.5 Detect problems early

More emphasis should be placed on early problem detection in projects with a large number of modules. Experienced designers should validate high level design. All vendor tools and new operating systems should be thoroughly evaluated and their performance tested and documented by the development team. Reliance on existing product documentation to assess the capability of vendor tools or operating systems leads to problems later on in the projects that require considerable time and effort to correct.

Developers should be encouraged to self-test and correct their code. This helps detect semantic errors in the code earlier in the development process.

7.1.6 Implement a corporate level strategy to deal with increases in software complexity

Complex problems should be decomposed into simpler and more understandable chunks. More effort should be spent in design partitioning to ensure that functionality is contained within modules. By containing the functionality within the modules, the need for communication between developers of different modules is greatly reduced. It limits the complexity as the need for interfacing between modules is reduced and also increases the number of modules that can be developed independently and simultaneously.

Complex project should be delivered in stages. Ensure that the most essential portion of the product is successfully in operation. Features and functionality should be planned for and added in a phased manner. As Gall (1986) states "A complex system that works is invariably found to have evolved from a simple system that worked... A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a simple working system".

A corporate level strategy to deal with complexity should exist. Such a strategy should include the selection of analysis and design methodology that can handle higher levels of abstraction, selection of higher level programming languages and Computer Aided Design tools, and an architecture evolution plan.

7.1.7 Verify assumptions

Managers should ensure that schedule estimates are made based on sound assumptions. In estimating development schedule, managers should consider the following:

1. the actual percentage of the programmer's work-week actually devoted to developmental activities
2. the extent and nature of the inter-dependencies between the various modules of a project
3. the number of sequential tasks in the project

4. the extra time required for planning, documentation, testing, systems integration and training that may be required for larger projects.

7.1.8 Build high performance software development teams

There is a need to improve the performance of software development teams. High performance software development teams are established with the objective of ensuring that software development is completed according to schedule. High performance software development teams should receive adequate support from senior management. Development resources and experienced personnel should be allocated to the team. Senior management should also arbitrate between departments to resolve problems between departments and the high performance software development team.

7.2 SUGGESTIONS FOR FURTHER RESEARCH

While considerable research effort has been dedicated to cost estimation in software development and maintenance projects (Banker, Datar, Kemerer and Zweig 1993, Andersen 1990, Tauseworthe 1981, Putnam and Fitzsimmons 1979, Putnam 1978, Herd, Postak, Russell and Stewart 1977, and Wolverton 1974), schedule estimation has received little attention.

The following suggestions for research questions that can be examined in future studies are offered:

- 1. What is the effect of factors such as newness of the problem domain and ambiguity of project goals on the task difficulty of software development projects?**
- 2. Do the most important reasons for schedule overruns found in this study also contribute to schedule overruns in non-software development projects?**
- 3. At what stage in the development process should schedule estimates be made? How should these estimates be made? How and when should the estimates be revised?**
- 4. How effective are prototypes in reducing changes in specifications?**
- 5. How effective are the following techniques in improving the development team's knowledge of the problem domain: consulting, hiring domain expert, locating designers at customer sites, and face-to-face meetings between designers and key-customers?**
- 6. What methods are available to help managers deal with changes in specifications after specifications are frozen?**
- 7. To what extent will self-testing reduce the number of errors and improve early error detection?**
- 8. How effective is the use of object-oriented (O-O) concepts in helping reduce the distribution of functionality between modules?**

9. What is the role of incentive structure in deciding the magnitude of schedule overruns?
10. How do product characteristics and process characteristics affect schedule overruns?

7.3 LIMITATIONS OF THE STUDY

This study has the following limitations:

In addition to complexity, novelty and ambiguity contribute to the ill-structuredness and task difficulty of a problem (Simon 1973, Mintzberg, Duru and Theoret 1976). Future research should study the impact on schedule overruns of novelty factors such as the newness of the problem domain and newness of technology and ambiguity factors such as ill-defined project goals, and uncertainty about the performance of vendor tools.

Future research should take into account the differences in complexity that result from the use of different programming languages (Kearney, Sedlmeyer, Thompson, Gray and Adler 1986).

This study captures the opinions of managers and senior software engineers on the reasons for schedule overruns in software development projects. It is likely that the data in this study embodies memory lapses and myopia of single respondents that are usually associated with retrospective, single respondent surveys. Since the respondents volunteered to provide data,

self-selection criterion might be a factor in the data. Also, the perspective of managers may be different from that of the senior software engineers. As only one respondent per project was allowed to provide data and as no distinction was made between the responses from managers and senior software engineers, the differences in perspective were not accounted for.

This study asked managers for opinions on constructs and did not measure the variables. For instance, managers were asked to rate the importance of lack of experience of the team as a cause of schedule overruns. This study did not measure experience of the team and correlate it to magnitude of schedule overruns.

Self-serving bias, attributing positive outcomes to internal factors and negative outcomes to external factors, may also be a factor in the responses.

BIBLIOGRAPHY

Abdel-Hamid, T.K., Sengupta, K. and Hardebeck, M.J. (1994), "The effects of reward structures on allocating shared staff resources among interdependent software projects: An experimental investigation" *IEEE Transactions on Engineering Management*, May, Volume 41, No. 2, pp. 115-125.

Adler, P.S., Mandelbaum, A., Nguyen, V. and Schwerer, E. (1995), "From project to process management: an empirically-based framework for analyzing product development time," *Management Science*, March, Vol. 41, No. 3, pp. 458-484.

Aggarwal, K.K. (1993), "Delay related reliability - A new performance index for computer-communication network dynamics" *Microelectronics and Reliability*, May, Vol. 33, No. 6, pp. 803-807.

Ancona, D.G. and Caldwell, D.F. (1990), "Beyond boundary spanning: Managing external dependence in product development teams" *Journal of High Technology Management Research*, No. 1, pp. 119-135.

Ancona, D.G. and Caldwell, D.F. (1992a), "Demography and design: Predictors of new product team performance" *Organization Science*, No. 3, pp. 321-341.

Ancona, D.G. and Caldwell, D.F. (1992b), "Bridging the boundary: External process and performance in organizational teams" *Administrative Science Quarterly*, No. 37, pp. 634-665.

Andersen, O. (1990), "The use of software engineering data in support of project management" *Software Engineering Journal*, Vol. 5, 6, November, pp. 250-256.

Balletti, A.J. and Guild, P.D. (1991), "Designers' impressions of direct contact between product designers and champions of innovation" *Journal of Product Innovation Management*, No. 8, pp. 91-103.

Banker, R.D., Datar, S.M., Kemerer, C.F. and Zweig, D. (1993), "Software complexity and maintenance costs" *Communications of the ACM*, November, vol. 36, No. 11, pp. 81-94.

- Basili, V.R.** (1980), "Quantitative software complexity models: A panel summary" *V.R. Basili Ed., Tutorial on Models and Methods for Software Management and Engineering*, IEEE Computer Society Press, Los Alamitos, California.
- Basili, V.R. and Hutchens, D.H.** (1983), "An empirical study of a syntactic complexity family" *IEEE Transactions on Software Engineering*, SE-9, 6, pp. 664-672.
- Basili, V.R. and Perricone, B.T.**(1984), "Software errors and complexity: an empirical investigation" *Computing Practices*, January, Volume 27, pp. 42-52.
- Bennatan, E.M.** (1992), "On time, within budget: Software Project Management Practices and Techniques" McGraw Hill International (UK) Limited.
- Boehm, B.W.** (1984), "Software engineering economics" *IEEE Transactions on software engineering*, Vol. SE-10, No. 1, pp. 4-21.
- Booch, G.** (1994), "Object-oriented analysis and design with applications" 2nd Edition, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California.
- Boydston, R.E.** (1984), "Programming cost estimate: Is it reasonable?" *Proceedings of the Seventh International Conference on Software Engineering*, pp. 153-159.
- Brooks, F.P. Jr.** (1975), "The Mythical Man-Month" Addison-Wesley.
- Brooks, F.P. Jr.** (1986), "No Silver Bullet: Essence and Accidents of Software Engineering" *Information Processing*, Elsevier Science Publishers B.V. North Holland.
- Brown, S.L. and Eisenhardt, K.M.** (1995), "Product development: past research, present findings and future directions" *Academy of Management Review*, Vol. 20, No. 2, pp. 343-378.
- Card, D.N., Page, G.T. and McGarry, F.E.** (1985) "Criteria for software modularization" *Proceedings of the eighth international conference on software engineering*, pp. 372-377.

Carmel, E. and Becker, S. (1995), "A process model for packaged software development" *IEEE Transactions on Engineering Management*, February, Volume 42, No. 1, pp. 50-61.

Chen, R.L. (1985), "Computerized life cycle advising, monitoring and predicting (CLAMP)" Ph. D dissertation, Texas A&M University, College Station, December.

Clark, K.B., Chew, W.B. and Fujimoto, T. (1987), "Product development in the world auto industry" *Brookings Papers on Economic Activity*, No. 3, pp. 729-781.

Clark, K.B. and Fujimoto, T. (1991) "Product development performance" Harvard Business School Press, Boston.

Clark, K.B. and Fujimoto, T. (1991b), "Product Development Performance: Strategy, Organization and Management in the World Auto Industry" MA: Harvard Business School Press.

Conte, S.D., Dunsmore, H.E. and Shen, V.Y. (1986), "Software Engineering Metrics and Models" Benjamin-Cummings, Reading, Mass.

Cooper, R.G. and Kleinschmidt, E.J. (1987), "New products: What separates winners from losers? *Journal of Product Innovation Management*, No. 4, pp. 169-184.

Curtis, B., Sheppard, S.B., Milliman, P., Borst, M.A. and Love, T. (1979), "Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics" *IEEE Transactions in Software Engineering*, SE-5, 2, pp. 96-104.

Curtis, B., Sheppard, S.B. and Milliman, P. (1979), "Third time charm: Stronger prediction of programmer performance by software complexity metrics" *IEEE*, pp. 356-360.

Dougherty, D. (1990), "Understanding new markets for new products" *Strategic Management Journal*, No. 11, pp. 59-78.

Dougherty, D. (1992), "Interpretive barriers to successful innovation in large firms" *Organization Science*, No. 3, pp. 179-202.

Dougherty, D. and Corse, S.M. (In press), "When it comes to product innovation, what is so "bad" about "bureaucracy"? *Journal of High Technology Management*.

Eisenhardt, K.M. and Tabrizi, B. (1995), "Accelerating adaptive processes: Product innovation in the global computer industry" *Administrative Science Quarterly*, Vol. 40, pp. 84-110.

Fortune, (1989), "Speeding new ideas to market" February 13, pp. 54-59.

Gall, J. (1986), "Systemantics: How systems really work and how they fail", Second Edition, The General Systemantics Press, Ann Arbor, MI., pp. 65.

Gibbs, W.W. (1994), "Software's Chronic Crisis" *Scientific American*, September, pp. 86-95.

Gupta, A.K. and Wilemon, D.L. (1990), "Accelerating the Development of Technology-Based New Products" *California Management Review*, Winter, pp. 24-44.

Hayes, R.H., Wheelwright, S.C. and Clark, K. (1988), "Dynamic manufacturing" Free Press, New York.

Herd, J.R., Postak, J.N., Russell, W.E. and Stewart, K.R. (1977), "Software cost estimation study-Study results" Doty Associates Inc., Rockville, MD, Final Technical Report, RADC-TR-77-220, Vol. 1, June.

Holtzblatt, K. and Beyer, H. (1993), "Making customer-centered design work for teams," *Communications of the ACM*, October, Vol. 36, No. 10, pp. 93-103.

Humphrey, W.S. (1988), "Characterizing the software process: A maturity framework" *IEEE Software*, March, Vol. 5, No. 2, pp. 73-79.

Iansiti, M. (1992), "Science-based product development: An empirical study of the mainframe computer industry" Working paper, Harvard Business School, Cambridge, MA.

Iansiti, M. (1993), "Real-world R&D: Jumping the product generation gap" *Harvard Business Review*, 71(3), pp. 138-147.

Jenkins, M.A., Naumann, J.D. and Wetherbe, J.C. (1984), "Empirical Investigation of Systems Development Practices and Results" *Information & Management*, 7, pp. 73-82.

Kearney, J.K., Sedlmeyer, R.L., Thompson, W.B., Gray, M.A. and Adler, M.A. (1986), "Software complexity measurement" *Communications of the ACM*, November, Volume 29, Number 11, pp. 1044-1050.

Korson, T.D. and Vaishnavi, V.K. (1986), "An empirical study of the effects of modularity on program modifiability" *Empirical studies of programmers*, Editors: E. Soloway and S. Iyengar, Ablex, Norwood, N.J.

Lind, R.K. and Vairavan, K. (1989), "An experimental investigation of software metrics and their relationship to software development effort" *IEEE Transactions in software Engineering*, vol. 15, No. 5, May, pp. 649-653.

Maldique, M.A. and Zirger, B.J. (1984), "A study of success and failure in product innovation: The case of the U.S. electronics industry, *IEEE Transactions in Engineering Management*, No. 4, pp. 192-203.

Maldique, M.A. and Zirger, B.J. (1985), "The new product learning cycle" *Research Policy*, 14, pp. 299-313.

Mintzberg, H., Duru, R. and Theoret, A. (1976), "The structure of unstructured decision processes" *Administrative Science Quarterly*, No. 21, pp. 246-275.

Munson, J.C. and Koshgoftaar, T.M. (1989), "The dimensionality of program complexity" *Proceedings of the International Conference on Software Engineering*, pp. 245-253.

Nobeoka, K. and Cusumano, M.A. (1992), "Multi-Project Strategy and Organizational Coordination in automobile Product Development" *Working Paper # 3487-92/BPS*, MIT Sloan School of Management.

Paulish, J.D. and Carleton, A.D. (1994), "Case Studies of Software-Process-Improvement Measurement" *IEEE Computer*, September, pp. 50-57.

Phan, D., Vogel, D. and Nunamaker, J. (1988), "The search for perfect project management" *Computerworld*, September, pp. 95-100.

Pressman, R.S. (1982), "Software engineering - a practitioner's approach" McGraw-Hill Book Company, New York.

Putnam, L.H. (1978) "A general empirical solution to the macro software sizing and estimating problem" *IEEE Transactions in Software Engineering*, July, pp. 345-361.

Putnam, L.H. and Fitzsimmons, A. (1979) "Estimating software costs" *Datamation*, pp. 189-198 September, continued in pp. 171-178 October, and pp. 137-140 November.

Robbins, S.P. (1993), "Organizational behavior concepts, controversies and applications" Sixth Edition, Prentice Hall, Inc., Englewood Cliffs, New Jersey, pp. 139-140.

Rowen, R.B. (1990), "Software project management under incomplete and ambiguous specifications" *IEEE Transactions on software engineering*, Vol. 37, No. 1, pp. 10-21.

Sackman, H., Erikson, W.J. and Grant, E.E. (1968), "Exploratory experimentation studies comparing online and offline programming performance" *Communications of the ACM*, 11, pp. 3-11.

Shen, V.Y., Yu, S., Thebaut, S. and Paulsen, L. (1985), "Identifying error prone software - An empirical study" *IEEE Transactions on software engineering*, Vol. 11, April.

Simmons, D.B. (1992), "A win-win metric based software management approach" *IEEE Transactions on Engineering Management*, February, Volume 39, No. 1, pp. 32-41.

Simmons, D.B., Ellis, C.N., and Escamilla, T.D. (1993), "Manager Associate" *IEEE Transactions on Knowledge and Data Engineering*, June, Vol. 5, No. 3, pp. 426-438.

Simon, H.A. (1973), "The structure of ill structured problems" *Artificial Intelligence*, No. 4, 181-201.

Soloway, E. and Ehrlich, K. (1984), "Empirical studies of programming knowledge" *IEEE Transactions in Software Engineering*, SE-10, 5 September, pp. 595-608.

Tausworthe, R.C. (1981), "Deep space network software cost estimation model" Jet propulsion lab, Pasadena, CA.

Thamhain, H.J. and Wilemon, D.L. (1986), "Criteria for Controlling Projects according to plan" *Project Management Journal*, June, pp. 75-81.

The Ottawa Citizen, (1995a), "Government cancels payroll contract" Saturday, April 22, pp.-2.

The Ottawa Citizen, (1995b), "Bill soars for air traffic system" Tuesday, May 9, pp. 1-2.

The Ottawa Citizen, (1995c), "Air traffic control automation a nightmare of mega-proportions" Sunday, May 14, pp. A1-A2.

United States House of Representatives Report, (1989), Committee on Government Operations, "DoD automated information systems experience runaway costs and years of schedule delays while providing little capability" pp. 101-382.

van Genuchten, M. (1991), "Why is Software Late? An Empirical Study of Reasons For Delay in Software Development" *IEEE Transactions on Software Engineering*, June, Vol. 17, No. 6, pp. 582-590.

Wolverton, R.W. (1974), "The cost of developing large-scale software" *IEEE Transactions on Computing*, June, pp. 615-636.

Workman Jr., J.P. (1995), "Engineering's Interactions with Marketing Groups in an Engineering-Driven Organization," *IEEE Transactions on Engineering Management*, May, Vol. 42, No. 2, pp. 129-139.

Zirger, B.J. and Maldique, M.A. (1990), "A model of New Product Development: An Empirical Test" *Management Science*, Vol. 36, No. 4. pp. 867-883.

Appendix - A

COVER LETTER

Date

Dear _____

I am a graduate student at Carleton University, working on a masters thesis titled "Reasons for schedule overruns in software development projects". The thesis is being supervised by Professor A.J. Bailetti of the School of Business. I seek your cooperation in providing the data for this study. Your name was suggested by _____.

The enclosed questionnaire takes up no more than 15 minutes to complete. It does not require the disclosure of sensitive data, nevertheless, I assure you that all data will be kept confidential. No link will be made between you and the data provided in this questionnaire and information from individual responses will not be divulged to anyone under any circumstances. Your name and phone number will enable me contact you if a questionnaire is returned incomplete or if clarification is required. You may fax the completed questionnaire to 788-2370. This is a secure and confidential fax. Access is restricted to persons authorized by Prof. A.J. Bailetti.

Information is sought on a software development project that you were intimately involved with, which experienced a schedule overrun and which was completed in the last two years. While every attempt has been made to make this questionnaire respondent-friendly, should you require any clarification, please call (613) 788-2370 or send an E-mail message to 'sgopalan@business.carleton.ca'. I shall address it promptly.

If you are interested in the results of the study please provide a mailing address in the space provided for this purpose at the end of the questionnaire. I appreciate your cooperation and highly value your input. Thank you in advance for promptly returning the completed questionnaire.

Sincerely,

Subramani Gopalan

**APPENDIX B
QUESTIONNAIRE**

Section 1: Project data

1.1) Your Name: _____ Phone Number: (____) _____

1.2) Company name: _____

1.3) The main line of business of your company is _____

1.4) Title of project: _____

1.5) Date of project completion: ____ / ____ / ____ (dd / mm / yy)

1.6) What is the appropriate description of this project? (Please tick the most appropriate category-choose only one).

Custom software development	☐
Embedded software development	☐
Software development for a target market	☐
Other _____	

Section 2: Complexity data

2.1.a) What is the total number of Source Lines of Code written for this project including new and reused code? : _____

2.1.b) Please specify the language(s) in which the code was written:

2.2) _____
What is the total number of modules developed for this project? :

2.3) _____
How does this project compare in size in relation to the other projects your company undertakes? (Please tick the appropriate category).

Very Large	Large	Average	Small	Very Small
☐	☐	☐	☐	☐

Section 3: Schedule overrun data

3.1) Please indicate the following for this project

	Estimated	Actual
Duration ¹ of project		

¹ Duration of a project is the time, in calendar weeks, from the start of high-level design to first customer delivery

3.2) Please indicate the impact of the schedule overrun on the following by circling the appropriate number:

	Very Significant			Not at all significant	
Project cost	1	2	3	4	5
Customer satisfaction	1	2	3	4	5
Progress of dependent projects	1	2	3	4	5
Financial success	1	2	3	4	5
Other _____	1	2	3	4	5
_____	1	2	3	4	5
_____	1	2	3	4	5

3.3) Please circle the importance of the following reasons in causing schedule overruns in this project. Circle N/A (Not Applicable) if the reason played no part in causing the schedule overruns:

		Not at all important			Very important		
1)	Changes in specifications	N/A	1	2	3	4	5
2)	Lack of senior management support	N/A	1	2	3	4	5
3)	Inadequate resource allocation	N/A	1	2	3	4	5
4)	Poor project management	N/A	1	2	3	4	5
5)	Incorrect estimation of development time	N/A	1	2	3	4	5
6)	Problems with organizing project teams	N/A	1	2	3	4	5
7)	Inability to detect problems early	N/A	1	2	3	4	5
8)	Lack of inter-departmental						

	cooperation	N/A	1	2	3	4	5
9)	Inflated user expectations	N/A	1	2	3	4	5
10)	Inadequate techniques	N/A	1	2	3	4	5
11)	Lack of experience of the team	N/A	1	2	3	4	5
12)	Other _____		1	2	3	4	5

- 3.4) If you had to do this project again, what would you do differently to reduce or eliminate the schedule overruns? (Please attach an extra sheet if required).

Section 4: Request for a copy of the findings

Should you be interested in receiving a copy of the findings, please provide your e-mail address here: _____

**Appendix C
CONSENT LETTER**

Date _____

Dear _____,

I am a graduate student in the School of Business at Carleton University, working on my masters thesis titled "Reasons for schedule overruns in software development projects" which is being supervised by Prof. Antonio J. Bailetti. I seek your help in identifying persons in your company who will provide data for my thesis research.

Respondents would be project leaders or senior software engineers who were associated with a software development project that was completed in the last two years and experienced schedule overruns. A brief questionnaire will be sent to the persons identified by you. The questionnaires will take no more than 15 minutes to complete. No sensitive information is sought and I assure you that all information will remain confidential.

In the space provided below please identify the persons who may provide data for this study. Please specify the number of employees at this location. You may fax your reply to 788-2370. This fax is confidential and is accessible only to persons authorized by Prof. A. J. Bailetti. If you have any questions please call either Professor Bailetti or me at 788-2370.

Approximate number of employees at this location: _____

Name of respondent: _____

Contact : (Ph) _____ - _____ (Fax) _____ - _____ E-Mail : _____

Name of respondent: _____

Contact : (Ph) _____ - _____ (Fax) _____ - _____ E-Mail : _____

Name of respondent: _____

Contact : (Ph) _____ - _____ (Fax) _____ - _____ E-Mail : _____

Name of respondent: _____

Contact : (Ph) _____ - _____ (Fax) _____ - _____ E-Mail : _____

Yours sincerely,

Subramani Gopalan
Graduate Student

Appendix D

CODING SCHEME

Section 1: Project data

- 1.1) Your Name: _____ Phone Number: (____) ____ - ____
1.2) Company Name: _____
1.3) The main line of business of your company is _____
1.4) Title of project: _____
1.5) Date of project completion: __ / __ / __ (dd/mm/yy)
1.6) What is the appropriate description of this project? Please tick the appropriate category.

Custom software development	1
Embedded software development	2
Software development for a target market	3
Other _____	4

Section 2: Complexity data

- 2.1) What is the total number of Source Lines of Code written for this project including new and reused code : _____
2.1.a) Please specify the language in which the code was written: _____
2.2) What is the total number of modules developed for this project? : _____
2.3.) How does this project compare in size in relation to the other projects your company undertakes? Please tick the appropriate category.

Very Large	Large	Average	Small	Very Small
5	4	3	2	1

Section 3: Schedule overrun data

3.1) Please indicate the following for this project

	Estimated	Actual
Duration ¹ of project		

¹ Duration of a project is the time, in calendar weeks, from the start of high-level design to first customer delivery

3.2) Please indicate the significance of the schedule overruns on:

	Very Significant			Not at all significant	
Project cost	5	4	3	2	1
Customer satisfaction	5	4	3	2	1
Progress of dependent projects	5	4	3	2	1
Financial success	5	4	3	2	1

3.3) Please circle the importance of the following reasons in causing schedule overruns in this project. Circle N/A (Not Applicable) if the reason played no part in causing the schedule overruns:

	N/A	Not at all important			Very important		
1) Changes in specifications	0	1	2	3	4	5	
2) Lack of senior management support	0	1	2	3	4	5	
3) Inadequate resource allocation	0	1	2	3	4	5	
4) Poor project management	0	1	2	3	4	5	
5) Incorrect estimation of development time	0	1	2	3	4	5	
6) Problems with organizing project teams	0	1	2	3	4	5	
7) Inability to detect problems early	0	1	2	3	4	5	
8) Lack of inter-departmental cooperation	0	1	2	3	4	5	
9) Inflated user expectations	0	1	2	3	4	5	
10) Inadequate techniques	0	1	2	3	4	5	
11) Lack of experience of the team	0	1	2	3	4	5	

12) Other _____ 1 2 3 4 5

3.4) If you had to do this project again, what would you do differently to reduce or eliminate the schedule overruns? (Please use extra sheets if required).

Section 4: Request for a copy of the findings

Should you be interested in receiving a copy of the findings, please provide your e-mail address here: _____

Appendix E

Spearman Rank Order Correlation Among the Importance of Reasons for Schedule Overruns

Reasons for schedule overruns	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
(R1) Changes in specifications										
(R2) Lack of senior management support	.1127 (.483)									
(R3) Inadequate resource allocation	.0297 (.854)	.3223 (.040)								
(R4) Poor project management	-.0217 (.893)	.4054 (.009)	.4081 (.008)							
(R5) Incorrect estimation of development time	-.1583 (.323)	.0055 (.973)	.4252 (.006)	.1351 (.400)						
(R6) Problems with organizing project teams	.0891 (.580)	.1856 (.245)	.3828 (.013)	.2659 (.093)	.1165 (.468)					
(R7) Inability to detect problems early	-.0547 (.734)	.0350 (.828)	.1611 (.314)	.3289 (.036)	.3092 (.049)	.0523 (.745)				
(R8) Lack of inter-departmental cooperation	.2384 (.133)	.3709 (.017)	.2549 (.108)	.2104 (.187)	-.0392 (.808)	.2854 (.070)	.1864 (.243)			
(R9) Inflated user expectation	.1835 (.251)	.3639 (.019)	.1132 (.481)	.4482 (.003)	-.0720 (.655)	.2686 (.089)	-.0448 (.781)	.0066 (.967)		
(R10) Inadequate techniques	-.1237 (.441)	.2053 (.198)	.0800 (.619)	.4189 (.006)	-.0629 (.696)	.1279 (.425)	.2896 (.066)	.2923 (.064)	.0779 (.628)	
(R11) Lack of experience of the team	-.0528 (.743)	.3422 (.029)	.1477 (.357)	.1909 (.232)	-.0804 (.617)	-.2284 (.151)	.0300 (.852)	.2858 (.076)	.1189 (.459)	.2919 (.064)

The significance levels are given in parentheses.

Appendix F

Spearman Rank Order Correlation Between the Importance of Reasons for Schedule Overruns and Size of Project Relative to the Size of Other Projects Normally Undertaken by the Company

Reasons for schedule overruns	Relative size of project
Changes in specifications	-.1128 (.483)
Lack of senior management support	.1674 (.296)
Inadequate resource allocation	.2945 (.062)
Poor project management	.1413 (.378)
Incorrect estimation of development time	.2023 (.205)
Problems with organizing project teams	.0318 (.843)
Inability to detect problems early	.0693 (.667)
Lack of inter-departmental cooperation	.0020 (.990)
Inflated user expectation	-.0611 (.704)
Inadequate techniques	.0135 (.933)
Lack of experience of the team	.0328 (.839)

Significance levels are given within parentheses.

Appendix G
Spearman Rank Order Correlation Between the Consequences of Schedule Overruns and the Importance of Reasons for Schedule Overruns

Reasons for schedule overruns	Project cost	Customer satisfaction	Financial success	Progress of dependent projects
Changes in specifications	.0699 (.664)	.0187 (.908)	.0227 (.891)	-.1447 (.367)
Lack of senior management support	.2609 (.099)	.1510 (.346)	.3980 (.012)	.0544 (.735)
Inadequate resource allocation	.0854 (.596)	.3540 (.023)	.0697 (.673)	.2979 (.059)
Poor project management	.4769 (.002)	.0900 (.576)	.3622 (.023)	.2820 (.074)
Incorrect estimation of development time	.3857 (.013)	.1557 (.331)	.1029 (.533)	.3156 (.044)
Problems with organizing project teams	.0463 (.774)	.2617 (.098)	.0861 (.602)	.2037 (.202)
Inability to detect problems early	.2590 (.102)	.0199 (.901)	.3088 (.056)	.2616 (.098)
Lack of inter-departmental cooperation	.1221 (.447)	.1029 (.522)	.1605 (.329)	.1742 (.276)
Inflated user expectation	.2547 (.108)	.1152 (.473)	.2078 (.204)	-.0184 (.909)
Inadequate techniques	.1172 (.465)	.0563 (.726)	.2455 (.132)	.1050 (.514)
Lack of experience of the team	.0328 (.838)	.1005 (.532)	.4252 (.007)	-.0033 (.984)

Significance levels are given in parentheses.

END

0 5 0 6 9 6

FIN