An investigation of the effects of computer-aided software engineering tools, system complexity, and system analyst's experience on system design quality and productivity: A laboratory experiment

Ongkasuwan, Metta, Ph.D.

Georgia State University - College of Business Administration, 1991

AN INVESTIGATION OF THE EFFECTS OF COMPUTER-AIDED SOFTWARE ENGINEERING TOOLS, SYSTEM COMPLEXITY, AND SYSTEM ANALYST'S EXPERIENCE ON SYSTEM DESIGN QUALITY AND PRODUCTIVITY: A LABORATORY EXPERIMENT

BY

METTA ONGKASUWAN

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree
of
Doctoral of Philosophy
in the
College of Business Administration
of
Georgia State University

GEORGIA STATE UNIVERSITY
COLLEGE OF BUSINESS ADMINISTRATION
DEPARTMENT OF COMPUTER INFORMATION SYSTEMS
1991

ACCEPTANCE

This dissertation was prepared under the direction of the candidates's Dissertation Committee. It has been approved and accepted by all members of that committee, and it has been accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Business Administration in the College of Business Administration of Georgia State University.

[illegible]
Dean
College of Business Administration

Dissertation Committee:

[illegible]
Chairperson

[illegible]
Member

[illegible]
Member

[illegible]
Member

ACKNOWLEDGEMENTS

A doctoral research is a challenge and required support and interest from individuals, faculties, colleagues, and friends. This dissertation is a result of the collective efforts of many people in searching for better understanding of interested issues over a period of time. An exceptional recognition and thanks go to Dr. Kuldeep Kumar, chairman of my dissertation committee, for his outstanding dedication and contribution throughout my doctoral research at the Computer Information Systems Department, College of Business Administration, Georgia State University, Atlanta, Georgia. Additional recognition and thanks also go to Drs. Bikramjit Garcha, Connie Wells, Charles Williams, Vijay Vaishanavi, Gordon Howells, and Karen Loch for their genuine interest and support throughout the course of this dissertation.

My doctoral research would not be complete without the sincere contribution from my colleagues and friends from IBM Corporation, KnowledgeWare, AT&T, U.S. Department of State of Georgia, California State University at Sacramento, University of California at Davis, Georgia State University, and Kennessaw College.

Finally, I am thankful to my parents and indebted to Chairat for their love, patient and endless support throughout my doctoral study and research in the United States.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

AN INVESTIGATION OF THE EFFECTS OF COMPUTER-AIDED SOFTWARE ENGINEERING TOOLS, SYSTEM COMPLEXITY, AND SYSTEM ANALYST'S EXPERIENCE ON THE SYSTEM DESIGN QUALITY AND PRODUCTIVITY: A LABORATORY EXPERIMENT

By

METTA ONGKASUWAN

October, 1991

Committee Chairman: Dr. Kuldeep Kumar

Major Department: Computer Information Systems

The primary purpose of this research is to investigate the effects of the use of Computer-Aided Software Engineering (CASE) tools on the syntactical quality of the system design specifications and productivity of the syntactic verification tasks. These effects are investigated under varying levels of system complexity and system analyst's experience.

In this research, a controlled laboratory experiment using both non-professional and professional system analysts as subjects was conducted to achieve the primary purpose. Multivariate Analysis of Variance (MANOVA), series of Analysis of Variance (ANOVA), and pair-wise t test were used to quantitatively analyze the experimental data. Protocol analysis of the experimental tasks was used to qualitatively analyze and explain the quantitative results.

The major findings from this study are summarized as follows. First, the use of CASE tool provides lower quality and productivity than traditional tool (paper and pencil). Second, if CASE tool is used as intended by its developer, it provides better quality and productivity than when it is used in the same manner as traditional tool. However, CASE tool still provides lower quality and productivity performance than traditional tool even when it is used as intended. The problem of poor performance of CASE tool seems to lie in the way each feature of CASE tool is used (e.g., difficult to use and connect information). Finally, system complexity and system analyst's experience do not seem to affect the quality and productivity of the use of CASE tool.

CHAPTER I

INTRODUCTION

The issue of computer-based information system development has evoked considerable interest and attention from both MIS managers and researchers. A survey conducted by Hartog and Herbert (1986) identified system development as one of the five most important issues facing MIS managers. Several MIS researchers (Acly, 1988; Bubenko, 1986; Jeffery, 1987; Olle, Hagelstein, Macdonald, Rolland, Sol, Asche & Verrijn-Stuart, 1986) also report that a "software crisis" exists in many industries. This crisis is characterized by problems of low system design quality, low system designer productivity, and ineffective management of the system development process. Two reasons are suggested for this crisis: an increased demand for better software quality, and an increased pressure for MIS organizations to improve system development productivity. Studies by Alloway and Quillard (1983) and Konsynski (1984) report problems of increasing software development backlogs. A survey by Konsynski in early 1980s discovered that these backlogs range from one to four years. As a result, a major challenge for MIS organizations is to find more efficient and effective methodologies and tools for improving system development productivity and quality.

A variety of system development methodologies have been proposed as the means for increasing the effectiveness of system development process. These methodologies provide guidelines for progressing through various phases in system development life cycle (see Table 1) (Davis & Olson, 1984). Examples of these methodologies include:

- Structured analysis and design methodology (DeMarco, 1978; Gane & Sarson, 1979; and Yourdon, 1989);
- Data-oriented methodologies (Warnier, 1981; Martin, 1988);
- Socio-technical system methodology (Mumford, 1981); and
- Decision-oriented development methodology (Keen & Scott-Morton, 1978).

These methodologies employ a variety of tools to model, represent, analyze, and specify the system under development. They also provide a variety of rules that assist system analysts in verifying system models or representations for correctness, consistency and completeness.

Two key problems have limited the effectiveness of most of these methodologies: the excessive paperwork required for system specifications, and the complexity of system development tasks. As system specifications evolve through a series of iterations, a large amount of effort is needed to keep and maintain a correct and complete set of specifications for the system. The time and cost to develop and maintain a complete and correct set of specifications have been found by some

Table 1

Stages and Phases of Systems Development Life Cycle

Stages in life cycle	Phases in life cycles	Description
Definition:	Proposal definition	Preparation of request for a proposed application.
	Feasibility assessment	Evaluation of feasibility and cost/benefit of proposed application.
	Information requirements analysis	Determination of information needed.
	Conceptual design	User-oriented design of application.
Development:	Physical design	Detailed design of data flows and processes in application processing system and preparation of program specifications.
	Physical database design	Design of internal schema for data in database or design of files.
	Program development	Coding and testing of computer programs.
	Procedure development	Design of procedures and preparation of user instructions.
Installation and Operations:	Conversion	Final system test and conversion.
	Operation & maintenance	Day-to-day operation, modification, and maintenance.
	Post audit	Evaluation of development process, application system, and results of use.

authors as too difficult to justify (Chickosky, 1988; Demarco, 1987; Yourdon, 1978).

The quest for greater productivity and quality in the current information systems development practices has led to the introduction of computer-assisted system development tools. These tools are expected to make it practical and economical to use these methodologies effectively in the development and maintenance of complete and correct set of system specifications.

These tools, which are commonly known as "Computer-Aided Software Engineering" or "Computer-Aided System Engineering" (CASE) tools, provide system analysts with computer support for developing and validating system specifications. In general, these tools support system development activities throughout system development life cycle (Newman, 1982; Konsynski, et al. 1984; Hoffnagle & Beregi, 1985; McClure, 1989; Martin, 1988). These tools also provide system analysts with powerful diagnostic features designed to ensure consistency, completeness and correctness of system specifications. Other features of these tools include: graphical features which help build and modify graphical representation of the systems (e.g., data flow diagrams, entity-relationship diagrams and other logical data models), data dictionary which store description of system components, prototyping and project management features.

CASE tools are being considered by MIS managers as a panacea for their current system development problems. Many companies selling CASE tools are advertising that their users are experiencing substantial improvement in system development productivity and quality. However, though many organizations are currently experimenting with these tools, only a few have fully implemented these tools (Bubenko, 1986). Furthermore, Betts and Suydam (1987) estimate that only 40% of purchasers continue to use these tools. Despite this dismal record of use, industry analysts predict that the market for CASE tools will continue to grow significantly in the 1990s. Therefore, it is important to assess the effectiveness and impact of these tools on the information system development process.

Focus of the Research

The introduction of CASE tools indicates a number of new avenues for information system research. Table 2 presents a sample list of research issues suggested by leading MIS researchers and practitioners (Norman & Nunamaker, 1989; Acly, 1988; Chikofsky & Rubenstein, 1988).

The focus of this research is on investigation of the effect of CASE tools on productivity and quality of system development process.

There are two reasons for this focus. First, due to the quality and productivity implications on the current software crisis and system development backlog, the management of quality and productivity of system development process is a challenge to MIS managers. Second, there seems to be a dearth

Table 2

A List of Research Questions and Issues Relating to CASE Tools

Applicability of CASE Tools:

1. What concerns motivate companies to adopt CASE tools and to what extent do the reasons for adoption have a bearing on the success of the adoption? What alternative system development tools are considered?
2. How can CASE users be characterized with regard to company size, organizational culture, competitive situation, type of products, MIS strategy, etc.?
3. What attributes characterize companies that have attempted to implement CASE tools, but, in their view, failed? What do their experiences reveal about applicability of CASE tools?
4. How many CASE applications are there in the industry, and at what rate are they being adopted?

Justification of CASE Tools:

5. Can a generic framework for justifying the adoption of CASE tools be developed?
6. What factors should be included in the justification of CASE tools? How should these factors be measured?
7. What are appropriate guidelines for making comparisons of CASE tools and their alternatives?
8. What are the sources of costs and benefits directly linked to the use of CASE tools?
9. **To what degree does the adoption of CASE tools lead to improved system development productivity and quality and the company's competitiveness?**

Implementation of CASE Tools:

10. How do we define successful and unsuccessful implementations of CASE tools?
11. Which factors are critical and which factors are non-critical for successful implementations of CASE tools?
12. How to measure the performance of CASE tools?
13. What types of system analysts' skills and training are essential for successful implementations of CASE tools?
14. To what degree do job classifications and job descriptions, evaluation and reward systems, and personnel selection procedures need to change in connection with CASE tools?

research literature that either supports or disputes productivity or quality claims advanced by the advocates of CASE tools.

In this study, the investigation focuses on the effect of the use of CASE tools on quality of the system specifications and productivity of the verification tasks during the requirement specification phase. The verification task investigated is a subset of verification activities--called "syntactic verification". Syntactic verification task is defined as the verification of internal consistency, correctness, and completeness of the system specifications. As the CASE tool used in this investigation relies on the structured analysis and design methodology (DeMarco, 1978; Gane & Sarson, 1978), the internal consistency based upon the requirements of structured analysis is defined as consistency and continuity of naming, defining and numbering of all enumerated elements in the system specifications. Similarly in the context of structured analysis, the syntactic correctness is defined as accuracy and precision of naming, defining and numbering of all elements in the system specification. Finally, syntactic completeness is defined in the sense that all enumerated elements are named, defined and numbered in the system specifications. Note, due to human intention based nature of requirements, no system development tools (with or without computer assisted) can ensure semantic and pragmatic completeness and correctness in the sense that all of the users' requirements have been met (Fraser, Kumar, &

Vaishnavi, 1991). This investigation is limited to syntactic verification only.

The primary reason for focusing on the syntactic verification tasks is the importance of these tasks for maintaining quality of the system specifications. These tasks ensure internal consistency, syntactical correctness, and syntactical completeness of the system specifications. If verification criteria are not met, the system specifications may be implemented with errors and further corrective actions at later stages of system development life cycle would be needed. Consequently, the syntactic verification tasks ensure overall quality of the system specifications.

Additional pragmatic reason for focusing on syntactic quality of the system specifications (instead of semantic quality of the design specifications) is that the current generation of CASE tools focus primarily on syntactic verification and do not provide automated support for semantic interpretation and verification of the system specifications. Therefore, a focus on syntactic verification tasks would examine one of the primary benefits attributed to the use of CASE tools.

In this research, the productivity and quality benefits of CASE tools are examined in the context of the structured analysis and design methodology as described by DeMarco (1978) and Gane and Sarson (1978). There are three reasons for using a CASE tool based upon the structured analysis and design methodology. First, it is a popular, easy to use, and commercially widespread methodology. Second, it provides a set of syntactic rules for validating the system specifications. These rules help ensure the consistency, completeness, and correctness of the system specifications. Third, it is the basis of many currently popular CASE tools including the tool used in this study.

Finally some authors (Benbasat & Vessey, 1980; Boehm, 1984; McCabe & Butler, 1989; Norman & Nunamaker, 1989, 1988; Fickas & Nagarajan, 1988) suggest that the productivity and quality of the system development process are also affected by the experience of the system analyst and the complexity of the system under development. Therefore, the productivity and quality impacts of CASE tools may be moderated by varying levels of system analyst's experience and system complexity.

In summary, the purpose of this research is to investigate the impacts of CASE tools on the productivity and quality of the syntactic verification tasks under varying levels of system analyst's experience and system complexity.

Objectives of the Research

The objectives of this research are to:

(1) test if the use of system development tools, system complexity, and system analysts' experience have significant effects on syntactical quality of the design specifications and on the productivity of the syntactic verification tasks;

(2) compare the performance of the use of CASE and traditional (paper and pencil) tools with respect to syntactical quality of the design specifications and productivity of the syntactic verification tasks; and

(3) examine the effects of different levels of system complexity (simple versus complex system) and system analysts' experience (less versus more experienced analysts) on the syntactical quality and productivity performance of the use of CASE and traditional tools.

Organization of the Dissertation

This dissertation is organized into seven chapters as follows.

Chapter I presents the introduction, focus, and objectives of this research.

Chapter II reviews and discusses previous literature and research relevant to this research.

Chapter III presents the research framework, research model, and research questions to be tested.

Chapter IV describes the research methodology, experimental design, subjects, tasks, procedures and variables, data collection methods, and statistical analysis methods used in this study.

Chapter V presents and discusses the statistical analysis of the experimental data and its results.

Chapter VI presents and discusses the protocol analysis of the data obtained from direct observations and post-experimental interviews and its results. These results reveal the system analysts' attitude toward system development tools and the different ways they perform syntactical verification tasks during the experiments.

Finally, Chapter VII provides the conclusion, summary of major findings, limitations, and implications of this research. Suggestions for further research are also presented.

CHAPTER II

LITERATURE REVIEW

The purpose of this chapter is to review and discuss previous literature and research relevant to this study. This literature review is limited to the literature and research associated with:

- computer-aided software engineering (CASE);
- system development productivity;
- system development quality;
- system analyst's experience and its impact on the system development process; and
- software, program, and system complexity.

Literature on Computer-Aided Software Engineering

The system development process is a complex process which is difficult to perform and manage (Langefors, 1973; Welke, 1983). The complexity involved in designing a large and complex information system would normally exceed the system analyst's capability to handle all necessary design tasks (Davis & Olson, 1984). Langefors (1973) has introduced the term "imperceivable system" to represent a system that has a very high number of parts and interactions such that its structure cannot be perceived or observed at one and the same time. Bubenko (1986) has suggested that automated system design tools may be used to facilitate the system analysts in performing necessary design tasks required in designing a large, complex, and imperceivable system.

In general, system design tools may be classified into two broad types: manual tools and computer-assisted (automated) tools. Over the past years, system analysts have used manual tools (e.g., paper and pencil) together with structuring aids (e.g., dataflow diagrams and data dictionary) to structure their thinking process. Recently, a new set of automated tools, known as "Computer-Aided Software Engineering" or "Computer-Aided System Engineering" (CASE), have been introduced and adopted by system analysts in many organizations.

CASE tools

In a broad sense, CASE may be viewed as a system development philosophy emphasizing an automation of either the parts of, or, the entire system development life cycle (Wasserman et al., 1982; Konsynski, et al. 1984). Some examples of the automated system design tools that support the design and development of large information systems are PLEXSYS, KnowledgeWare, Excelerator, Design/1 and PSL/PSA.

CASE tools may be classified further into two broad categories: front-end CASE (or upper CASE) tools and back-end CASE (or lower CASE) tools. Front-end CASE tools support system development activities in the early phases of the system development life cycle (e.g., information requirements analysis and conceptual design) whereas back-end CASE tools support the translation of system specifications into programming code (McClure, 1989; Martin, 1988). Examples of Front-end CASE tools are "Excelerator" by Index Technologies, "Design Aid" by Nastec, and "IEW Analysis and Design" by KnowledgeWare. Examples of Back-end CASE tools include the "Microstep" programmer workbench by Syscorp Information and the "POSE" by Computer Systems Advisers. Some tools such as IEW workbench may include features of both Front-end and Back-end CASE tools.

CASE tools provide a variety of functions in support of system design and development tasks. These functions include: diagramming, error checking, data repository management, prototyping, code-generation, re-engineering of usable codes, methodology reinforcement, graphical representation, and target system selection (McClure, 1989). Table 3 summarizes some of the common functions of CASE tools and enumerates examples of features which support such functions.

<u>Strengths and Weaknesses of CASE Tools</u>

CASE tools currently available in the market represent early stages of evolution of CASE products. Many CASE developers still continue to improve their CASE products. At the current state of CASE tools, like any other system design

Table 3

Functions and Features of CASE Tools

Function	Feature
Diagramming	Data Flow Diagram, Structure Chart, Decision Table/Matrix, Entity-relationship diagram.
Error checking	Checking Syntax, Consistency, Completeness, Traceability.
Data repository management	Host-based (Encyclopedia), PC-based (Data Dictionary), DBMS.
Prototyping	Simulation, Functional Model, Screen Painter, Report Painter.
Code-generation	Skeleton Program, Complete Program, On-line or Batch Programs.
Re-engineering of usable codes	Static Analyzer, Re-documentation, Restructuring, Reverse Engineering, Dynamic Analyzer.
Methodology reenforcement	Structure Analysis and design by Demacro, Yourdon, Gane/Sarson, Warrnier-Orr; Information Engineering by James Martin; and Object-Oriented Design.
Graphical representation	Color, Window, Use of Mouse.
Target system selection	On-line or Batch System, Transaction Processing System, Real-time System, Embedded System.
Interfaces	Planning, Analysis, Design, Implementation, Maintenance, and Project management.

and development tools, they have both strengths and weaknesses. Table 4 provides a summary of the strengths and weaknesses of CASE tools reported by numerous MIS practitioners and researchers (Chikofsky, 1988; Connor & Case, 1986; Corkery, 1986; Margolis, 1988; Marcus & Nelson, 1989).

Previous Research on the Impacts of CASE Tools on the Productivity and Quality of System Development

The introduction of CASE tools in the mid 1980s created new avenues for MIS research. Table 5 provides a brief summary of the current CASE research found in IS literature. Most of the research related to CASE tools is still exploratory in nature and employs such research methodologies as case studies and field surveys.

Norman and Nunamaker (1989) conducted a survey of ninety-one MIS managers from forty-seven organizations using CASE products. The purpose of their study was to investigate the system analysts perception regarding:

- their preferences of specific CASE product components related to productivity;
- if there was any productivity improvement in the communication process during information systems development when CASE was employed;
- whether there was productivity improvement in coherence with enterprise system development standards when CASE was used; and

Table 4

Strengths and Weakness of CASE Tools

CASE Strengths:

- Improving software quality
- Reducing development time and cost, hence, increase productivity
- Enforcing software/system engineering standard
- Making prototyping and structured analysis technique practical
- Enabling reuse of software components (e.g., prototypes, data, system and program architectures, program and data structure designs, data models, and programming codes)
- Simplifying programming maintenance
- Freeing developers to focus on creative part of software development
- Encouraging evolutional and incremental development
- Improving communication among developers

CASE Weaknesses:

- Relying on structured methodologies (SDLC)
- Lacking only a methodology support standard
- Having limited functions
- Not providing integrated central repository
- Not providing integrated interfaces/tools
- Requiring structured analysis and design methodology skills
- Supporting narrow scope of development activities
- Having long learning curve

Table 5

A Summary of Selected Research on the Impacts of CASE Tools on System Development Productivity and Quality

Reference	Research Method	Subjects	Independent Variables	Dependent Variables	Major Findings
Norman, Nunamaker (1989)	Survey	91 subjects know how to use CASE tools from 47 U.S. and Canada companies	CASE tools, standard, communication	Productivity	The dominance ranking reveals that DFD is the most time consuming. Completeness and consistency checks are the least important functions. Data model, E-R diagram have the most impact on productivity.
Orlikowski (1989)	Case study	Programmers, junior analysts, and senior managers from Beta Consulting Company	CASE tools, applications, developers	Organization structure	CASE tools have disrupted the social relation among project members. New skill is needed to ensure the reliability of system development in an organization.
Necco, Tsai, Holgenson (1989)	Survey	63 companies listed in the Directory of Top Executives	CASE tools	Productivity, quality of system design	Only 24% of 63 companies use system design CASE tools. They claim that CASE tools improve productivity, quality, and communication among users and developers.
Ryan (1989)	Survey	569 ID departments in U.S. firms	CASE tools	Cost, quality	Users do not expect cost saving from use of CASE tools. They expect quality improvement from CASE.

(Continued...)

Table 5 (Continued)

Reference	Research Method	Subjects	Independent Variables	Dependent Variables	Major Findings
Marcus, Nelson (1989)	Survey	40 programmers analysts, and designers who use CASE tools in 12 U.S. companies	CASE tools, project types, programming experience	Productivity	The improvement of productivity depends on suitability of projects, programming experience and tools.
Yellen (1990)	Experiment	31 juniors and senior students at University of North Texas	CASE tools, manual tools	Quality: - correctness - completeness - communicability	CASE tools are superior to manual tools in term of correctness. CASE tools do not help users to develop a complete system nor to understand problem better.

- the system analysts' perceptions of CASE product components with respect to their productivity.

The results from Norman and Nunamaker's study indicate that system analysts perceived:

- that they can identify, via their ordering preferences, the parts of CASE products that contribute the most to increase in their productivity over manual methods;
- their communication through CASE products, as opposed to non-CASE communication, does not make a significant impact on productivity;
- adherence to information system development standards when using CASE does not make a significant impact on productivity; and other
- data flow diagraming feature, data dictionary, project standardization, and screen and report design facilities contribute the most to productivity.

It was also found in Norman and Nunamaker's study that CASE tools improve productivity and deserve further attention and evaluation.

Orlikowski (1989) conducted a case study to investigate the effects of CASE tools on organizational structure and performance. The subjects were junior system analysts, programmers, and senior system analysts with and without CASE tool experience at the Beta Consulting Corporation. Five different application projects (four large projects and one small project) selected by senior managers were used in this study. An average of four weeks was spent on observing and interviewing one project at a time. Overall of one hundred and twenty interviews averaging one and a half hour per interview were conducted in this case study. The observations were made for both a CASE user group and a non-CASE user group throughout the system development life cycle. The findings from this study suggested that the introduction of CASE tools disrupts and changes social relationships among project team members in the organization. These changes include the division of labor and the pattern of dependency among team members. Orlikowski suggested that new specialized skills may be needed in order to ensure the quality (reliability) and productivity of the system development when using CASE tools.

Necco, Tsai, ánd Holgenson (1989) conducted a survey of sixty-three MIS organizations listed in the Directory of Top Computer Executives to determine the current usage of CASE tools. The results indicate that only twenty-four percent of the organizations that participated in the study have used CASE tools. They also reported that the perception of the organizations using CASE tools is that the productivity of system developers, quality of system design, and communication among system developers and users have improved. However, system developers do not believe that the use of CASE tools will make maintenance at the later phases easier.

Ryan (1989) conducted a survey of 569 information service departments to determine the amount of the IS budget allocated to adopting and implementing CASE in their organizations. Ryan's study found that an average IS annual budget in the organizations using CASE was $47.7 million, while the budget in the organizations not using CASE was $16.8 million. Ryan advocated that even though CASE users did not expect cost savings from the use of CASE tools, they expected that the use of CASE tools would be justified by higher quality and productivity which would require less and easier system maintenance.

Marcus and Nelson (1989) conducted a survey of forty programmers, system analysts, and designers who have used CASE tools at twelve companies. The results of this study indicate that the productivity improvement through the use of CASE is dependent on the suitability of the project and the developers' experience with CASE tools. Nelson and Marcus suggested that instead of spending time on program coding, developers should spend more time on planning, analyzing and designing the system through CASE tools.

Yellen (1990) conducted laboratory experiments to determine whether CASE tools are capable of improving the quality of the SDLC process and product. The subjects were thirty-one juniors and seniors enrolled in a university information systems curriculum. The subjects were carefully divided into two homogeneous subgroups according to age, sex, GPA, and real-world system analysis experience. The first group was assigned to use a CASE tool to prepare data flow diagrams and data dictionary entries while the second group was assigned to use a traditional pencil and paper-based method. The outputs of these two groups were then compared with respect to three attributes of quality (i.e.,correctness, completeness, and communicability). The experimental results indicated that correctness is the only attribute of quality in which a CASE tool surpasses a traditional paper and pencil tool. However, a CASE tool was found to support only those subjects who know how to perform the tasks either with or without a CASE tool.

In summary, the past studies on CASE tools, except for Yellen's study (1990), are survey-type studies that measure the respondents' perception on productivity and quality improvements due to CASE. In Yellen's study (1990), although a laboratory experiment was conducted to investigate the effect of CASE tools on quality of the system design process and product, the measurement of quality are subjective judgements. The results from these studies are inconclusive.

Very little is known about the effectiveness of CASE tools under various environmental factors such as system developer characteristics, organization characteristics, and allocation of resources in the organization. Thus, CASE is an area in which many research issues currently exist. These issues span a variety of concerns and may require multiple research methodologies to investigate.

Literature on System Development Productivity

This section reviews the literature and previous research that addresses definitions and measurements of system development productivity, and the variables that may affect system development productivity.

General Definitions and Measurements of System Development Productivity

Davis and Olson (1984) define productivity in information system development as productivity of system developers such as system analysts, programmers and knowledge workers. Davis and Olson, however, do not provide any instruments to measure the productivity of system developers.

Beruvides and Sumanth (1987) have defined productivity of system developers as the ratio of the sum of total tangible outputs to the sum of the total tangible inputs. It can be mathematically represented by the following equation:

$$TP = \frac{\theta}{I_H + I_T}$$

where:

TP	=	Total productivity
θ	=	Total tangible output
I_H	=	Human input
I_T	=	$I_M + I_C + I_E + I_X$
I_M	=	Material input
I_C	=	Capital input
I_E	=	Energy input
I_X	=	Other expenses

Beruvides and Sumanth have suggested that I_T, which is the sum of material, capital, energy and other expenses inputs, is relatively small when compared to the human input (salary). They have also suggested that the total productivity equation may be appropriate in estimating the productivity of the system developers if the total tangible output can be objectively measured.

QED Information Sciences Inc., Wellesley, Massachusetts (1989) has presented several measures for system development productivity. Some examples of these measures are man-hours; costs; ratio of outputs to inputs; professional system analyst performance measure; productivity changes over time; total value-added (e.g., ratio of labor value-added to labor cost); and comparison of productivity at industry level (e.g., competitive factors--market share, sales factors, asset factors, personnel factors, management factors, organizational factors, information technology factors). Some of these measures (e.g., ratio of outputs to inputs, man-hours, costs, system analyst performance measure, and value added) are quantitative measures and can be used to measure the system development productivity. QED has suggested that research is needed to investigate the system development productivity at the industry level in order to describe the productivity of system development across different industries. The findings may be useful in suggesting directions in which the current organizations should follow and maintain their productivity level in the industry. However, measures such as total value-added, and comparison of productivity at the industry level are not applicable to this investigation as the focus of our study is at the analyst level.

In summary, it was found in the current literature a majority of productivity measures is based upon the traditional concept of output/input ratio analysis. A major difficulty in using this ratio is that it is somewhat difficult to measure the absolute values of all inputs to, and the outputs from the system development process.

Variables Affecting System Development Productivity

Various researchers (Pietrasanta, 1980; Jeffery, 1987; Turner, 1987; Pressman, 1987) have suggested that several additional variables could influence the productivity of the information system development. These variables include:

- variables related to the application (e.g., size, complexity);
- environmental variables (e.g., requirements stability, interface controls, testing complexity);
- attributes of the system development process (e.g., planning, support tools, computer hardware support); and
- characteristics of the system developer (e.g., individual experience and capability, management capability).

A brief review of the literature summarizing factors or elements that may influence system design productivity was presented by Turner in 1987. In his review, Turner identified eight basic elements that may influence productivity of system design process. These elements include: system concepts (i.e., system need); system boundary (e.g., size and complexity of the system); division of system development tasks, system structure; decomposition of the system; operating sequence; performance measures; and extent of change. Turner suggested that further research is needed to verify the respective significance of each of these elements.

Putnum also developed and presented a cost model of system development productivity in 1987. Putnam's cost model includes development effort, time, labor-rate, and technology. The model can be represented by the following equation:

$$K = L^3/C_k{}^3 t_d{}^4$$

where:

K	=	Development effort (in person-years)
L	=	Lator-rate factor (\$/person-year)
C_k	=	State of technology constant
t_d	=	Development time (in years)

In the above model, the productivity is determined by the ratio of system size to the development effort (K). Therefore, if the system developers want to maintain the same level of productivity, and if they desire to decrease the development effort for the same size of project, they need to increase the time for the project.

In 1987, Boehm also developed and presented another cost model that can be used to predict the system development productivity in terms of efforts and time. Boehm's model can be mathematically represented by the following equation:

$$\text{Man-months} = 2.4(\text{KDSI})^{1.05}$$
$$\text{Elapsed time} = 2.5(\text{MM})^{0.38}$$

where:

KDSI	=	Thousands of delivered source instructions
MM	=	Man-months derived from Man-months equation

In Boehm's model, the productivity of system development is determined by the ratio of the system's size to man-months. Based on this model, for a given size of the system development project the smaller the effort, the greater the productivity of the system development is. Furthermore, Boehm's model has also incorporated the cost driver factors which can influence the value of size and effort in the productivity ratio. These cost drivers include: product attributes (e.g., database size, and product complexity); computer attributes (e.g., execution time, and main storage constraint); personnel attributes (e.g., system analyst capability, application experience, machine experience, and programming language experience); and project attributes (e.g., use of modern practices and tools, and schedule constraint).

Benbasat and Vessey presented an earlier framework for investigating the effects of system development factors on productivity in 1980. In Benbasat and Vessey's framework, the system development productivity is measured by total time to develop the system. The system development factors in their framework include:

- organizational operations characteristics (e.g., levels of user involvement, degree of group interactions, leadership style, and number of development standards);
- computer hardware characteristics (e.g., types of computer system used, size of computer memory, access speed);
- source languages (e.g., machine languages, natural languages, and high level of procedural languages);
- developer characteristics (e.g., experience, mathematical aptitude ability, and system development skills);
- problem characteristics (e.g., types of problem, complexity of problems, and types of data required);
- software engineering characteristics (e.g., methodologies, tools, techniques, and procedures); and
- programming mode characteristics (e.g., batch, and on-line).

In summary, the past literature on system development productivity provides some theoretical models of productivity. These models suggest several factors that may influence the productivity of system development process. These factors include organizational characteristics, development process characteristics, developer characteristics, user characteristics, system characteristics, and software engineering characteristics. However, further research is needed to investigate the effects of each of these factors on the development productivity under a variety of development conditions.

Literature on System Development Quality

This section reviews the literature pertaining to system development quality. Table 6 provides a brief summary of this literature. A review of selected literature follows.

Gilb (1977) has defined the system design quality as the conformance of the system design to the predetermined, explicitly stated functional and performance requirements. Gilb further proposes the metric concepts to define and measure quality of the system design. A metric is defined as a language for describing the set of attributes demanded in the target system specification (system design). Metrics are independent of the functional process or structure of the system. They can be used to describe the quality of the system without describing the functional details of subsystems. They allow a system designer to concentrate on user requirements. The set of metrics includes:

(1) Multidimensional Quality of System Design Metric. It is a metric that uses the following quality factors to describe and measure the quality of a system and its related subsystem: reliability, maintainability, availability, cost, implementation time and effort, and relationships among subsystem designs.

Table 6

A Summary of Literature Related to System Design Quality

Reference	Type of Literature	Major Findings
Gilb (1977)	Descriptive	The use of Metric Concepts to define and measure quality of system design: - System Attribute Specification - Multidimensional Quality of System Design Matrix (reliability, maintainability, availability, cost, implementation time and effort, speed, and relationships among system designs); - Dataware Metrics (Reliability metric, Maintainability metric, Accuracy metric, Flexibility metric, Structure metric, Performance metrics, Resource metrics, and Diverse metrics).
Jalote (1989)	Experimental (simulation)	Completeness is the most critical factor for system design quality. Automatic tools can be used to detect the incompleteness of system design.
McCall (1977)	Descriptive	Three aspects of system design quality include: operational, revision, and transition characteristics.
Olle, et al. (1986)	Descriptive	Three system design quality factors include: completeness, correctness, and consistency.
Yeh (1982)	Descriptive	Three factors affecting the system design include: communication, complexity, and evolvability.
QED (1989)	Descriptive	Factors affecting the system design may include: organization, system development, data processing operations, software, and data.
Pressman (1987)	Descriptive	Applied system design quality metrics: Halstead's software science model, McCabe's complexity model, and Review Checklist of system design quality.

(2) Dataware Metrics. The data description concepts (input codes, record design, and data bases) are used to describe the properties of the system design. The following metrics can be used to describe specific properties of data and measure the quality of the system design:

- reliability metric (e.g., error detection probability, error correction probability, repairability, security);
- maintainability metric (e.g., documentation, built-in diagnostic aids, automatic recovery procedures);
- accuracy metric (e.g., degree of freedom from error);
- flexibility metric (e.g., level of logical complexity, portability, redundancy and integrity);
- structure metric (e.g., redundancy ratio, depth or number of levels of hierarchy in the system designs, number of linkages, number of modules);
- performance metric (e.g., effectiveness which comprises of operational reliability, system readiness, design adequacy; efficiency which is the ratio of effectiveness to cost); and
- resource metric (e.g., financial ratio, total system cost, incremental cost, return on investment, and suggested man-year).

These metrics are discussed in detail in Gilb (1973). The metrics use simple concepts of quality which can be easily calculated and measured. The reliability metric suggested by Gilb (1977) can be used in this investigation to define and measure the quality of system design.

McCall, Richards, and Walters (1977) have proposed a quality model that can be used to define and measure the system design quality. This model focuses on three essential aspects of the design: operational characteristics, revision characteristics, and transition characteristics. The operational characteristics include:

- correctness (i.e., does the system design represent what the user wants?);
- reliability (i.e., does the system design have an accurate representation of the desired system at all time?);
- efficiency (i.e., will the new system design represent the system that works with the existing system?);
- integrity (i.e., is the system design secure?); and
- usability (i.e., can the system design be used?).

The revision characteristic include:

- flexibility (i.e., can the system design be changed and revised?);
- testability (i.e.,can we test the system design?); and

- maintainability (i.e.,can the system design be easily documented and maintained?).

The transition characteristics include:

- reusability (i.e., can the system design be reused?);
- portability (i.e., can we transform the system design to fit with different computer system?); and
- interoperability (i.e., can we integrate the new system design with other system designs?).

Olle, Hagelstein, MacDonald, Rolland, Sol, Van Asche, and Verrijn-Stuart (1986) have suggested that the system design quality can be determined by measuring three factors: completeness, correctness, and consistency. The completeness is subjectively judged by the users and developers. The correctness is defined as the ability to satisfy the given constraints of the system development, and subjectively measured by the users and the limitation of the system being developed. The consistency is defined as the extent to which the system design does not contain contradictions. The authors also suggest that formal representation techniques may be used to detect the contradictions and improve the quality of the system designs and assist in writing complete system design.

Jalote (1989) conducted a simulation study to test whether completeness is the most desirable property for system design. A VAX system running Unix was used to generate a set of test cases and test data to detect the incompleteness of system design. It was found in this study that if the system designs are not complete, the implementation will not be completed, and the behavior of all of the consequent operation can not be defined. Jalote has suggested that an automatic tool (simulation) can be used to detect the incompleteness of system design.

In summary, previous research and literature related to system design quality indicate that system design quality metrics (Gilb, 1973, 1977; McCall, et al. 1977) are available. However, most of these metrics have not been empirically tested and validated.

Literature on System Analyst's Experience

This section reviews the literature and previous research investigating system analyst's expertise and its impact on the system design quality and productivity. Table 7 presents a brief summary of the major findings of these research. A review of selected literature is provided below.

Boehm (1981) has included system analyst experience and capability in his COCOMO model. Several researchers have investigated the effect of programmer's experience on the programmer's productivity (Chrysler, 1978; Lucus & Kaplan, 1976; Thadhani, 1984). Their findings are inconclusive and do not suggest that programmer experience has a significant effect on productivity.

Table 7

A Summary of Literature Related to System Analysts' Experience

Reference	Research Methodology	Major Findings
Adelson (1984)	Experiment	Experts form abstract representation, novices form concrete representation. Novices surpass experts when dealing with concrete detailed-problems.
Adelson & Soloway (1985)	Experiment	Analyst experience in system analysis and design domain is gained and accumulated over time with familiar problem and lost when confronted with non-familiar problem.
Eilot (1985)	Case study	Expert analyst solve the problem from top-down, consistent, quality approach; Novice takes bottom up approach, influenced by analogy.
Grant-Mackay (1987)	Case study	Expert/novice problem solving behavior: expertise is lost when working with unfamiliar tools and problems.
Guindon & Curtis (1988)	Experiment	Identify cognitive process during Curtis software design and tools required to support each task.
Lewis & Sier (1983)	Experiment	Experts surpass novices in diagnostic of project failure.
Prieto-Diaz (1987)	Survey	Proposed domain analysis technique to capture domain activities and outcomes in a set of data flow diagram.
Simon (1981)	Simulation	Experts solve complex problems faster and more accurately than novices. Indexed node-link structures is used by experts to solve problems.
Sternberg & Davison (1982)	Experiment	Problem solving depends on individual intellectual ability, knowledge, motivation, style and execution process.
Vessey (1985)	Experiment	Novices surpass expert programmers in recalling program when debugging program and using noncongruent declarative knowledge.
Vitalari (1985)	Experiment	Identifies core knowledge utilized by system analyst.

Adelson and Soloway (1985) have defined the system analyst as a problem solving specialist who applies his or her system development experience to identify, analyze, evaluate and develop the user requirements and system specification. The system analyst expertise is defined as the level of knowledge and skills the system analyst gains and accumulates while performing the system development tasks over a given period of time. The system development knowledge can be gained through traditional classroom education, hands-on training, or solving real business problems over a period of time. Normally, it takes years for a system analyst to master a specific skill. The training instrument, procedures and time all have effects on the system analyst skills and the accumulation of the system development knowledge. Adelson and Soloway concluded that the highly skilled system analyst can develop a familiar system better than the one with less skill and with an unfamiliar system.

Vessey (1985) conducted an experiment to investigate the differences in the debugging processes of expert and novice programmers. Sixteen programmers from the State Government Computer Center, Brisbane, Queensland were the subjects who participated in the study. The set of instruments used to differentiate novice and expert programmers in the study include: peer rating and comprehensive tests (recall tests by Shneiderman, 1977; question-answer tests; synopses of program function by Weissman, 1974). Each subject was asked to reproduce the program either verbatim or a functionally equivalent version. A program was a short 67-line COBOL program. A verbal protocol method was used to collect and analyze the data. The results of this study indicate that the novice programmers surpass expert programmers in recalling the programs when debugging and using non-congruent declarative knowledge (detail knowledge).

Kolodner (1983) conducted a simulation study to compare novices' and experts' reasoning models. A computer simulation program called "SHRINK" was used to implement the theory of expertise. The findings of this study indicate that experts know more about their domain, and are able to apply and use that knowledge more effectively than novices.

Vitalari (1985) conducted a quasi-experimental study to describe the content of the system analyst's domain knowledge. Eighteen experienced system analysts were asked to solve an accounts receivable problem. A verbal protocol analysis was used to capture and analyze the data. It was found in this study that there are six types of knowledge utilized by system analysts: organization specific knowledge, functional domain knowledge, application domain knowledge, knowledge of techniques and methods, core system analysis domain knowledge, and high-rate knowledge. Vitalari suggests that further research is needed to investigate and describe system analysts' problem solving process.

Guindon and Curtis (1988) conducted an experiment using a verbal protocol method to describe the cognitive processes of professional system analysts during a software design process.

Three professional designers in the study were selected by their peers and managers as very skilled and competent. They were given two hours to produce a design solution that was in a form and at a level of detail that could be implemented by programmers. Videotapes were used to capture additional data which described the strategies and the breakdowns of the complex task in their problem solving processes. The findings of the study indicate that there are four main components of the system analysts's cognitive model of software design. The four main components are: knowledge sources (i.e., technology domain, application domain, problem domain, design schemes, design methods, concepts); design process control (i.e., design meta-schema, methods, heuristic, primary position); connected internal representation and processes (i.e., of problem domain and solution domain); and connected external representations and processes given available tools and media. Guindon and Curtis suggest that a set of aids or tools can be used to support the system analysts' cognitive processes in the design of a system or software. Examples of these aids or tools include a library of reusable design schemes, design journal, special displays of all constraints on the solution in order to augment working memory, and visual simulation tools.

In summary, review of the literature related to system analyst's expertise has led us to conclude that the system analyst's expertise has a significant impact on the output of the system design process. However, there are no effective instruments currently available which can help in measuring system analyst's expertise and performance.

Literature on System Complexity

Complexity in system development process can be discussed at two levels: software/program complexity and system complexity. Table 8 summarizes software/program complexity and system complexity measures found in the current literature.

Software and program complexity

Several measures for program complexity have been proposed in the literature. Examples are McCabe's Cyclomatic Complexity Metric (McCabe and Butler, 1989); Halstead's Programming Effort Metric (Halstead, 1977); Albrecht's Function Point Metric (Albrecht, 1979); and Oviedo's Data Flow Complexity Metric (Ovideo, 1980).

McCabe and Butler (1989) proposed the cyclomatic complexity metric as a way to measure program complexity. This can be determined by the following equation:

$$v = e - n + p$$

where:

v	=	Complexity of the program
e	=	Number of edges in a program flow graph

Table 8

Software, Program, and System Complexity Measures

Reference	Complexity Measure		
McCabe (1976)	Program Complexity	=	Number of nodes + Number of connections between nodes in the program
Halsted (1980)	Program Complexity	=	f(Number of operands, Number of operators in the software)
Oviedo (1980)	Program Complexity	=	Total number of data flow in the diagrams
Albrecht (1979)	Function Point: Size of Program	=	(Information processing size) x (Technical complexity factor) x (Environmental factors)
Langefors (1973)	System Complexity	=	f(Number of components, Number of interactions among components)
Welke (1983)	System Complexity	=	f(Cardinality and Variety of IS/DSS)
Simon (1981)	System Complexity	=	f(Number of parts in the system, Number of interaction of those parts)
Wright (1974)	System Complexity	=	Inverse proportion of Time pressure
McCabe & Butler (1989)	System Complexity	=	Summation of individual components design complexity in a structure chart
Konsynski (1984)	Structural Complexity	=	f((R1,R2,R3,R4) and (P1,P2,P3)) where R1, R2, R3, R4 denote each realm of system life cycle, and P denotes the properties of complexity in each realm (P1 = volume, P2 = distribution, P3 = location)

n	=	Number of nodes (vertices)
p	=	Number of connected components

Based on the mathematical properties of this model, McCabe defines the complexity of the program as the maximum number of linearly independent paths through the program.

Halsted (1977) has used the program volume and program level to determine the programming effort. Then, the programming effort can be used to identify the degree of program complexity. The programming effort can be determined by the following equation:

$$V = (n_1 \log_2 n_1 + n_2 \log_2 n_2) . \log_2 (n_1+n_2)$$

where:

V	=	Programming effort (a measure of software complexity)
n1	=	Number of distinct operators that appear in a program
n2	=	Number of distinct operands that appear in a program
N1	=	Total number of operator occurrences
N2	=	total number of operand occurrences

Halsted's programming effort model can be applied to measure the software complexity from the program volume point of view. However, it may not be applicable to measure the software structural (level) complexity.

Oviedo (1980) has proposed a data flow complexity model for measuring the program complexity. Data flow complexity of the program can be calculated by the following equation:

$$DF = \sum_{i=1}^{s} DF_i$$

where:

DF	=	Data flow complexity of a program body
s	=	Set of blocks in the program body

Oviedo's data flow complexity model can be applied to measure the complexity of the data flow diagram of the system specification.

Albrecht (1979) has suggested that the function point metric can be used to measure the size of programs and their associated programming effort. The size of the program is determined by the product of three factors: information processing size (e.g.,inputs, output, files, and inquiry); technical complexity factor (i.e., estimation of degree of influence of fourteen components of general application characteristics); and environmental factors (e.g.,risk, people skill, methods, tools, and language) (Symonds, 1988). However, only the first two factors are used to estimate the size of the program in the function point method. The third factor has not been taken into account in the correct version of function points as defined by Albrecht (1979). Further research is needed to describe and include the measurement of the environmental factors into the function point method. The function point method, then, could be used to measure the complexity of the system.

System complexity

Langefors (1973) and Simon (1981) have suggested that the system complexity is the association of the number of components in the system and the number of interactions among components in the system. The more components and interactions in the systems, the more complex the system appears.

Wright (1974), however, has argued that time and constraint are other elements that make up system complexity. By linking Langefors's and Simon's system complexity concepts (Langefors, 1973; Simon, 1981) and Wright's time and constraint elements, the comprehensive definition of system complexity becomes the association of the number of components in the system (e.g., processes, data flows, decisions, constraints) and their interactions within a given time frame. Therefore, the greater the number of processes and interfaces within a short period of time and high business pressure, the more complex the system.

Welke (1983) has suggested that system complexity can be measured by its cardinality (number of instances) and the variety of the information system development support systems.

Konsynski (1984) has introduced the structural complexity metric for measuring the complexity of system designs and estimating complexity between system development life cycle phases. The structural complexity metric partitions the system life cycle into four sets of complexity realms denoted by R1, R2, R3, and R4. The structural complexity metric can be represented by the following functions: $P = f(R1, R2, R3, R4)$. R1 deals with the complexity of the requirement specifications in a complete and consistent logical design.

R2 involves the complexity of designing, constructing, and implementing a physical design which is consistent with the logical design. R3 is concerned with the complexity of the overall operating efficiency of the target system. R4 deals with the flexibility in the system implementation.

The complexity measures in each realm are denoted by P, where P1 is equal to volume complexity (e.g., size of entity), P2 is equal to distribution complexity (e.g., interrelatedness), and P3 is equal to location complexity (e.g., intermodular interfaces). The complexity measures within a given realm can be used to project the complexity and productivity of realm activities.

McCabe and Butler (1989) have developed a system design complexity metric based upon his earlier work (McCabe, 1976). The system design complexity (S_0) of a design module M is defined as:

$$S_0 = \sum_{i \in D} iV(G_i)$$

where:

S_0	=	System design complexity
V	=	Cyclomatic complexity of each graph
i	=	Module number
G	=	Flow graph
D	=	Set of descendants of M modules

McCabe's System Design Complexity Metric can be used to estimate the time and effort for developing the system design specification. An automated tool for computing the system design complexity has been developed by McCabe and Associates.

McCabe's automated tool is now available and is being applied on several projects.

In summary, when a system becomes very large and integrated, system complexity can lead to severe coding and maintenance problems. Many researchers have attempted to develop techniques and mechanisms for estimating the complexity of a system. Further research is needed to verify these techniques and associate them with automated tools.

Summary

This chapter has reviewed previous research and literature relevant to this study. The review covers the following areas: Computer-Aided Software Engineering (CASE), system development productivity, system development quality, system analyst's expertise, and system complexity.

In this research, an attempt is made to integrate the research discussed above into a framework to determine and analyze the effects of certain system development variables on system development productivity and quality. Specifically, this research builds upon the work of Norman and Nunamaker (1989) on CASE; Vitalari (1985) on system analyst knowledge base; Fraser, Kumar, and Vaishnavi (1991) on syntactic verification of the specifications; Langefors (1973) on system complexity; Benbasat and Vessey (1980) on system development productivity; and Gilb (1977) and Miller (1989) on system development quality.

CHAPTER III

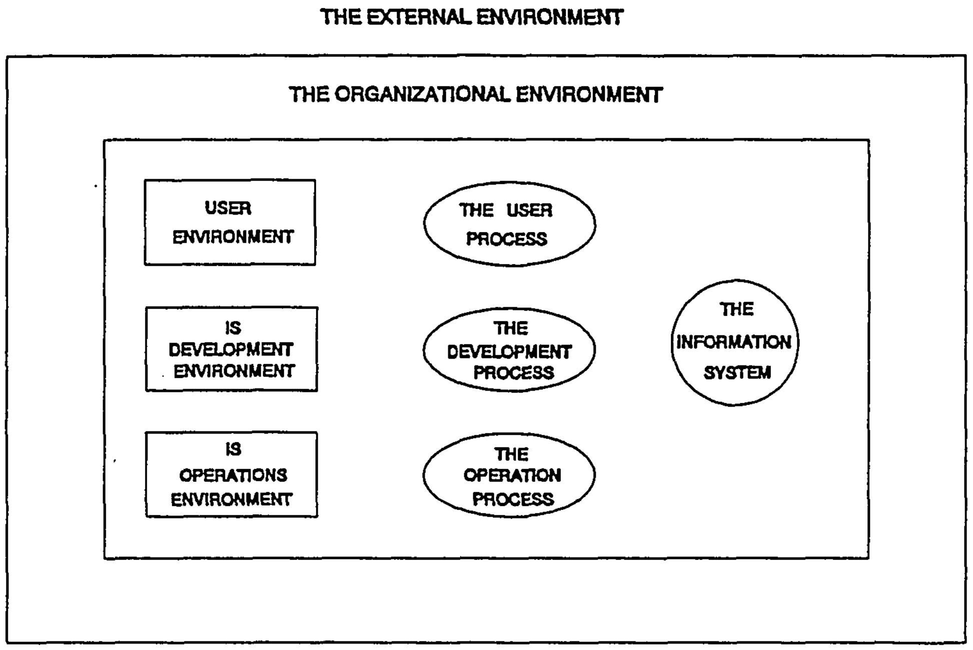
RESEARCH MODEL

The purpose of this chapter is to present the research framework, research model, research questions and hypotheses which form the basis for this study.

Research Framework

Research in MIS can be better understood when viewed in the context of a generic research framework such as the one proposed by Ives, Hamilton, and Davis (1980). Their framework identifies three general classes of variables that may affect the performance of a computer-based information system in an organization. These three classes of variables, shown in Figure 1, are environmental variables (shown as rectangles), process variables (shown as ellipses), and information system variable (shown as a circle).

The environmental variables include the external environment, the organizational environment, and the information system environment. The external environment variable refers to eight major factors that impact the performance of organization. These factors are legal, social, political, cultural, economic, educational, resource, and industry/trade factors. The case of organizational



Source: Ives, B., Hamilton, S., & Davis, G. B. (1980). "A Framework for Research in Computer-based Management Information Systems." Management Science, 26(9), p. 917.

Figure 1. General research framework

environment variables refer to the organization's goals, tasks, structure, volatility, and management philosophy or style. Finally, the information system environment is further subdivided into environmental variables, process variables, and information system variable. These variables are described below.

The environmental variables within the information system environment consist of the user environment, the IS development environment, and the IS operations environment. These three variables determine the type of IS to be developed, the development methodologies and personnel, and the critical resources required for the operation of information systems.

The process variables consist of three sub-variables: the development process, the operational process, and the user process. The development process, by selection and application of critical development resources (within environmental constraints), produces the information system. The operations process is the physical operation of the IS and is primarily a function of the operations resource. The user process, focusses on the usage of IS by the primary user, is usually measured by task accomplishment leading to an effect on the productivity and quality of decisions.

Finally, the information system variable consists of three classes of variables: IS content, presentation form, and time of presentation. The IS content refers to data and decision models available in the IS. The presentation form refers to method by which the information is presented to users. The time of presentation refers to reporting intervals, processing delays, and on-line or off-line data storage.

Research Model

The Ives, Hamilton, and Davis's model identifies a large set of variables for MIS research. This research focuses on a subset of these variables: the IS development environment, the IS characteristics, and the IS development process. Figure 2 graphically identifies the focus of this study.

The purpose of this study is to investigate the effect of IS development environment on IS development process and its product, i.e.,the resulting system. The IS development environment is characterized by the tools used in IS development process (traditional versus automated tools), the complexity of the resulting system being developed (simple versus complex), and the experience of system analyst in IS development process (less and more experience). In this study, the performance of IS development process is measured by its productivity and quality of its product. A model summarizing these variables is presented in Figure 3.

In summary, the abbreviated research model, presented in Figure 3, includes three components of the IS development environment (system design tools, system complexity, and

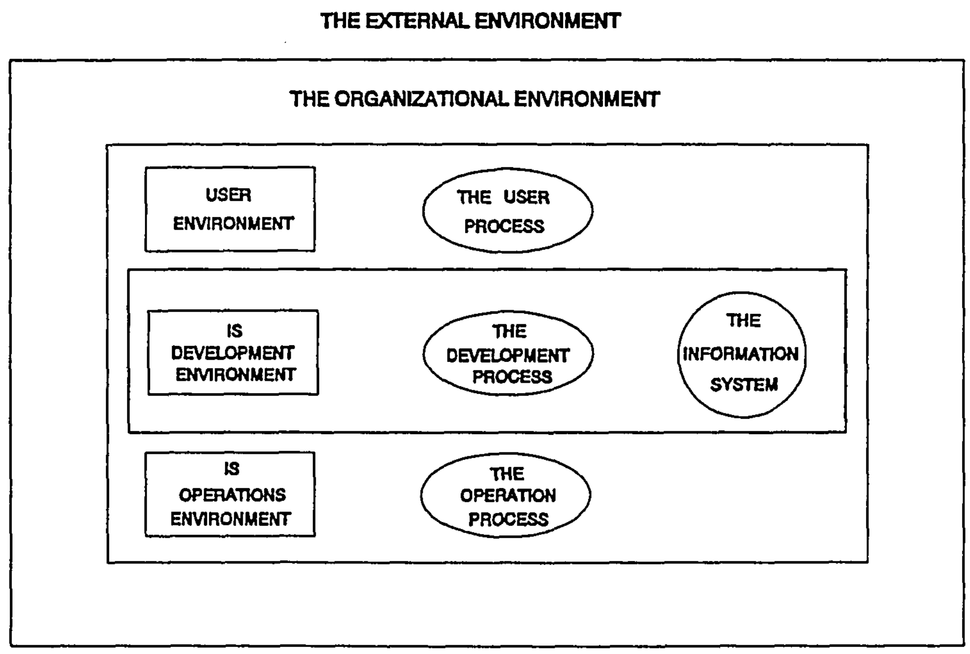


Figure 2. Specific research framework

system analyst's experience). The outcome variables (the development process and the resulting IS) are represented as productivity of the syntactic verification tasks and syntactic quality of the resulting system design specifications.

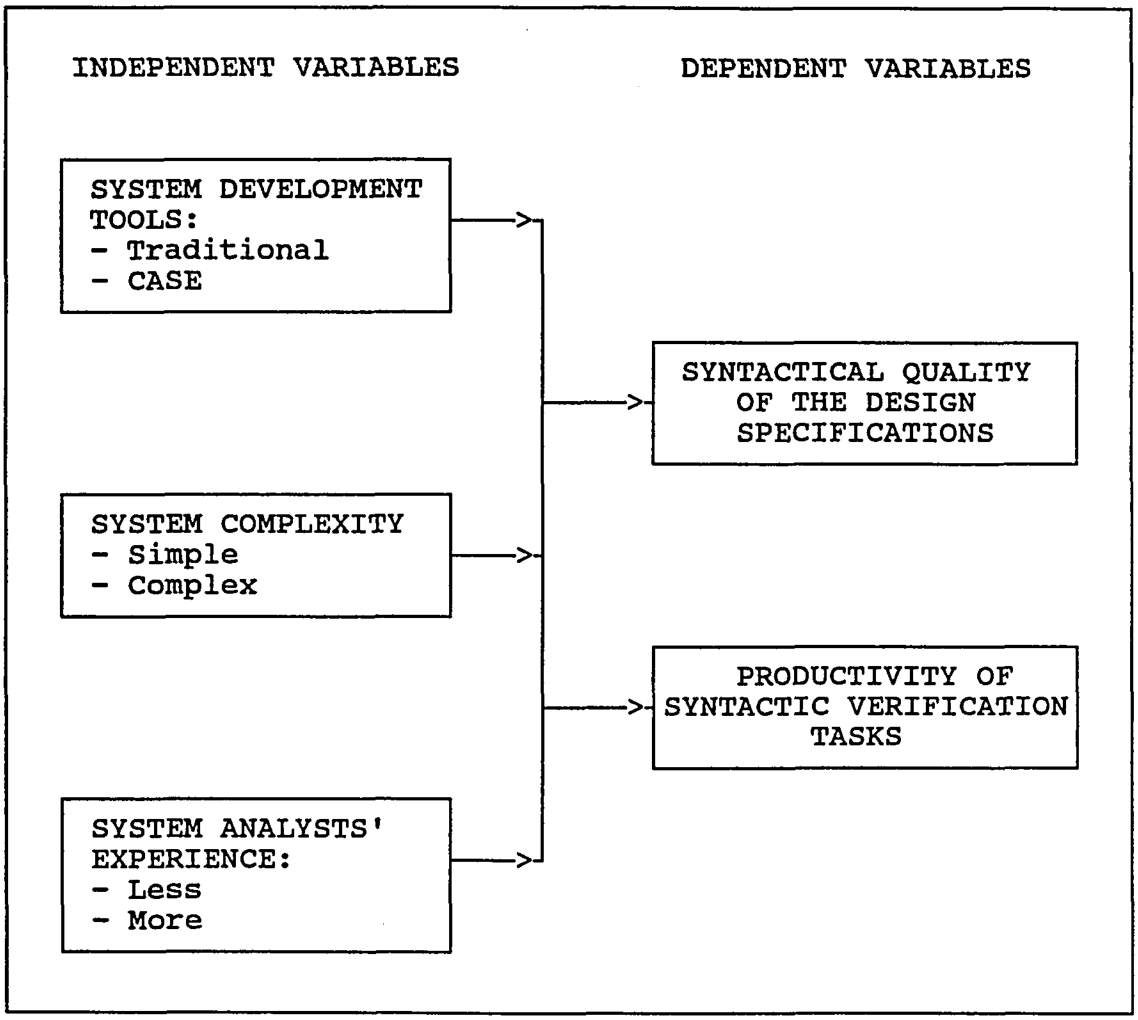
Model Variables

Independent Variables

The research model in Figure 3 identifies three independent variables. A brief description of each of these variables is provided below. A detailed definition, operation and measurement of these variables is discussed in Chapter IV.

The first independent variable represents the system development tools used in the verification process. The model examines the use of two types of system development tools: paper and pencil (traditional tool) and computer-assisted tool (CASE tool).

The second independent variable in the model is the complexity of the system under development. For the purpose of this study, system complexity is operationally defined as the cardinality (number of instances) of system components (Langefors, 1973; Welke, 1983). Two levels of system complexity are examined: simple and complex systems. For a simple system, the number of data flow diagrams, data


INDEPENDENT VARIABLES
DEPENDENT VARIABLES
SYSTEM DEVELOPMENT TOOLS:
- Traditional
- CASE
SYSTEM COMPLEXITY
- Simple
- Complex
SYSTEM ANALYSTS' EXPERIENCE:
- Less
- More
SYNTACTICAL QUALITY OF THE DESIGN SPECIFICATIONS
PRODUCTIVITY OF SYNTACTIC VERIFICATION TASKS

<u>Figure 3</u>. Research model

dictionary entries, and processes is much smaller than that in a complex one.

The third independent variable in the research model is the experience of system analyst performing the syntactic verification tasks. Two levels of system analyst's experience are considered: less experienced and more experienced system analysts. Less experienced analysts are those analysts who have lesser experience than more experienced analysts in terms of the number of years which they have been working as professional system analysts and the number of system design projects which they have completed using CASE tools.

Dependent Variables

The research model has two dependent variables: syntactical quality of the system design specifications, and productivity of the syntactic verification tasks. Syntactic verification tasks are defined as verification of internal consistency, syntactic correctness and syntactic completeness of the system design specifications (Fraser, Kumar, & Vaishnavi, 1991). Syntactical quality is defined as the degree to which the system design specifications are internally consistent and syntactically correct and complete. Productivity of the syntactic verification tasks is defined as the number of syntactical errors found and correctly changed per unit of time. Specific measures of syntactical quality and productivity in this study are discussed in Chapter IV.

Research Questions

The purpose of this research is to investigate the effects of system development tools (traditional versus CASE tools), system complexity (simple versus complex) and system analyst's experience (less versus more) on syntactical quality of the system design specifications and productivity of the syntactic verification tasks. Specifically, this research addresses three research questions.

Research Question 1:

> Do the use of CASE tools, system complexity, and system analysts' experience have significant effect on syntactical quality of the system design specifications and productivity of the syntactical verification tasks?

Various MIS researchers (McCabe & Butler, 1989; Norman & Nunamaker, 1989; Pressman, 1987; Benbasat & Vessey, 1980) suggested that types of system development tools, levels of system complexity, and levels of system analyst's experience influence quality and productivity of information system development. However, no research has been conducted to validate whether or not these factors have significant effect on syntactical quality of the system design specifications and productivity of the syntactical verification tasks. The first research question, therefore, is concerned with the testing of significance of effects of these factors. We hypothesize that system development tools, system complexity, and system analysts' experience have significant effects, both individually and interactively, on syntactical quality of the system design specifications and productivity of the syntactical verification tasks.

Research Question 2:

> Does one particular system development tool always outperform the other tool in terms of syntactical quality of the system design specifications and productivity of the syntactic verification tasks? If so, which tool is better? If not, what is the relative performance of the two tools for each combination of system complexity and system analysts' experience levels?

Many companies selling CASE products claim that users of their products have achieved substantial improvements on their system development quality and productivity. They report that CASE tools are more effective than traditional tools under all possible use environments. Various MIS practitioners and researchers (Chikofsky, 1988; Marcus & Nelson, 1989; Norman & Nunamaker, 1989; Orlikowski, 1989; Necco, Tsai, & Holgenson, 1989) suggest that the use of CASE tools can enforce system engineering standards and reduce time and cost required for developing a system. Hence, CASE tools are being considered as panacea tool for improving system development quality and productivity. The second research question, therefore, is concerned with the evaluation of relative performance of traditional and CASE tools. We hypothesize that the use of CASE tools do not outperform the use of traditional tools for all levels of system complexity and system analyst's experience; it is likely that different levels of system complexity and system analyst's experience affect the relative performance of the two tools. If the experimental results confirm this hypothesis, further analysis will be performed to examine the performance difference between the two tools under each combination of system complexity and system analyst's experience levels. This would lead us to the third research question stated below.

Research Question 3:

> How do different levels of system complexity and system analyst's experience affect performance of traditional and CASE tools in terms of syntactical quality of the system design specifications and productivity of the syntactic verification tasks?

The third research question is, therefore, concerned with the examination of effects of different levels of system complexity and system analyst's experience on the performance of traditional and CASE tools. To investigate this research question, additional data analyses were performed to identify changes in quality and productivity of the two tools with respect to changes in system complexity and system analysts' experience levels.

Summary

This chapter presents the research framework, model, questions and hypotheses to be tested in this study. The research model identifies three independent and two dependent variables. The independent variables consist of types of system development tools (traditional versus CASE tools) used by system analysts to perform the syntactic verification tasks, levels of system complexity (simple versus complex systems) whose specifications are to be verified, and levels of experience of system analyst (less versus more experienced) who performs syntactic verification tasks. The dependent variables consist of syntactical quality of the system design specifications and productivity of the syntactic verification tasks. The three research issues investigated in this study are: the significance of effects of system development tools, system complexity, and system analysts' experience on syntactical quality of the system design specifications and productivity of the syntactic verification tasks; the relative performance of traditional and CASE tools; and the effects of system complexity and system analyst's experience on performance of the two system development tools.

CHAPTER IV

RESEARCH METHODOLOGY

The purpose of this chapter is to describe the research methodology used in this study and to provide a description of the experiment design, subjects, tasks and procedures. This chapter also describes the operationalization of the research variables and their measurements. This description is followed by a discussion of the data collection and the statistical analysis techniques used to analyze experimental data.

Selection of the Research Methodology

A controlled laboratory experiment was used as the research methodology in this study. This particular research method was chosen because it allows the researcher to manipulate the variables of interest and control other variables which are not of main interest in the study. Laboratory experiments are powerful research methods that provide the researcher with the capability to discover and measure cause-and-effect relationships among variables. They also provide very high level of internal validity (Stone, 1978; Fromkin & Streufert, 1983).

Laboratory experiments, however, have at least one major drawback. They may suffer from a lack of external validity if unrealistic subjects and tasks are used in the experiment. To remedy against this potential problem, the subjects used in this study include both non-professional system analysts (students who are enrolled in an advanced system analysis and design courses in business schools) and professional system analysts (system analysts who have been working in industry as system analysts for at least five years). In addition, the task to be examined in this study is syntactic verification task which is a necessary task for ensuring the quality of system requirement specifications.

Experimental Design

A 2^3 factorial experimental design with three independent variables is used in this study (see Figure 4). The first independent variable is the type of system development tools used. This study investigates the performance of the use of two different types of system development tools: traditional paper-pencil based tools versus CASE tools. System complexity is the second independent variable in the experiment. This variable has two levels: simple versus complex systems. Finally, system analyst's experience is the third independent variable. This variable also has two levels: less experienced system analysts versus more experienced system analysts. Thus

	Less Complex System	More Complex System
Less Experienced System Analyst	Using Traditional Tools	Using Traditional Tools
	Using CASE Tools	Using CASE Tools
More Experienced System Analyst	Using Traditional Tools	Using Traditional Tools
	Using CASE Tools	Using CASE Tools

Figure 4. A schematic representation of experimental design

the experimental design has 2 x 2 x 2 or 8 cells (treatments). Following the recommendation of Wasserman and Kutner (1985) for each cell, four replications were conducted. Thus, the sample size used in this study is 8 x 4 or 32 subjects.

Experimental Subjects

The subjects participating in this research consist of both less and more experienced system analysts. Table 9 summarizes the qualification requirements for these two group of subjects. The descriptions of and recruiting procedures for the subjects are presented below.

Less Experienced System Analyst Subjects

Less experienced system analysts are those analysts who have knowledge of both system analysis and design method and general business functions, but lack real-world work experiences. In this study, less experienced system analyst subjects were recruited from system analysis and design students at Georgia State University (Atlanta, Georgia) and Kennessaw State College (Kennessaw, Georgia). To qualify as less experienced system analyst subjects, the students were required to have the following qualifications.

(1) Knowledge of structured analysis and design methods and techniques.

Table 9

A Summary of the Qualification Requirements for Less and More Experienced System Analyst Subjects

Qualification Requirements	Less Experienced System Analysts	More Experienced System Analysts
Knowledge of structured analysis methods and techniques *	Yes	Yes
Knowledge of general business functions **	Yes	Yes
Knowledge of CASE tools ***	Representative CASE tool	Representative CASE tool
Minimum of five years of work experience as system analyst, plus has completed at least four system analysis and design projects using the structured analysis method, and one project using CASE tools	No	Yes

Note: * Knowledge of data flow diagram, process description and data dictionary
** Knowledge of accounting, finance, marketing, and production and inventory management
*** Knowledge of a representative CASE tool used in this study

(2) Knowledge of general business functions (e.g., accounting, finance, marketing, and production and inventory management).

(3) Knowledge of the representative CASE tool. "The representative CASE tool" refers to as the CASE tool used in the experiment. This tool has been available commercially in the market since 1984. The less experienced subjects must have completed the CASE tool training and developed at least one system specification using the CASE tool.

(4) No actual work experienced as system analyst in the real business or non-profit organization.

The recruiting procedure for less experienced system analyst subjects included the following steps:

Step 1: The researcher contacted the instructors who taught the advanced system analysis and design course at Georgia State University and Kennessaw College. Care was taken to ensure that the instructors understood the nature and objective of the study and the contributions of the study to MIS research and practices.

Step 2: The students in the classes, whose instructors agreed to allow them to participate in the experiments, were requested to fill out a Subject Background Questionnaire (see Appendix A). The data from this questionnaire were used to determine if student met the minimum qualification requirements for less experienced system analyst subjects in this study.

Step 3: Students who met the minimum qualification requirements for less experienced system analysts were invited to participate in the experiment. They were told up front that there was no financial compensation offered. However, they were motivated to participate in the study by their interest in system analysis and design research and education.

More experienced system analyst subjects

More experienced system analysts are those who have both the knowledge and skills of system analysis and design, and work experience as system analysts in business organizations. To qualify as more experienced system analyst subjects for this study, the analysts were required to have the following qualifications:

(1) Knowledge of structured analysis and design methods and techniques.

(2) Knowledge of general business functions (e.g., accounting, finance, marketing, production and inventory management).

(3) Knowledge of the representative CASE tool.

(4) A minimum of five years of work experience as system analysts or system designers. During these five years, they must have completed at least four system analysis

and design projects using the structured analysis methods and at least one project using the representative CASE tool.

The recruiting procedure for more experienced system analyst subjects included the following steps:

Step 1: The researcher contacted MIS or system development managers of medium and large corporations located in the Atlanta, Sacramento, and San Francisco areas. Care was taken to ensure that the managers understood the nature and objective of the study, and the contributions of the study to MIS research and practices. They were then asked to refer the researcher to individual analysts who would agree to participate in the study.

Step 2: Each potential subject was contacted and a Subject Background Questionnaire (see Appendix A) was administered by phone during an initial screening call by the researcher. For each subject, the level of experience and knowledge of the structured analysis methods and techniques, general business functions, and CASE tools were determined.

Step 3: Only subjects who met the minimum qualification requirements for more experienced system analyst subjects stated in Table 9 were invited to participate in the experiments. As with the less experienced system analyst subjects, the qualified subjects were told up front that there was no financial compensation for participating in the experiment. However, they were motivated to participate in the study by their interest in system analysis and design research and education.

Experimental Tasks

As a part of the experimental procedure, subjects were asked to perform syntactic verification and correction of provided system specifications. These tasks involved determining if the design specifications were internally consistent (e.g., level balancing, numbering, and naming); correct (e.g., valid data flow directions, process names and numbers, file names, and external entity names); and syntactically complete (e.g., no missing element in the data flow diagrams, process descriptions, and data dictionary). The verification task included both diagnosis and correction of the system specification. The verification for semantic completeness and correctness of the design specifications, in the sense that all of the user's requirements are complete and correct, is not included in this study.

In the experiment, subjects were provided with a system specification problem case to diagnose and correct. The case provides the subjects with company background information and an initial set of system specifications in the form of data flow diagrams, process descriptions, and data dictionary entries. Subjects were asked to perform syntactic verification of the system design specifications using the rules of structured analysis methodology and the tool (either traditional paper-pencil based tool or CASE tool) given to them. In the process of doing so, they were asked to diagnose the initial design specifications and make all necessary corrections for internal consistency, correctness, and syntactical completeness.

Experimental Procedures

The experiment was administered individually to each subject (i.e., there was only one subject in each experimental session). Figure 5 outlines the major steps followed in each experimental session. These steps are described in detail as follows:

Step 1: The subject was briefed about the experimental tasks to be performed and the procedures to be followed during the experiment.

Step 2: The subject was requested to fill out the consent form (Appendix B). The purpose of the consent form was to secure subject's cooperation and to request the subject to keep the content of the experiment confidential.

Step 3: A short pilot session using a small example system specification problem (different from the ones used in the actual experiment) was given to each subject to acquaint him/her with the experimental tasks,

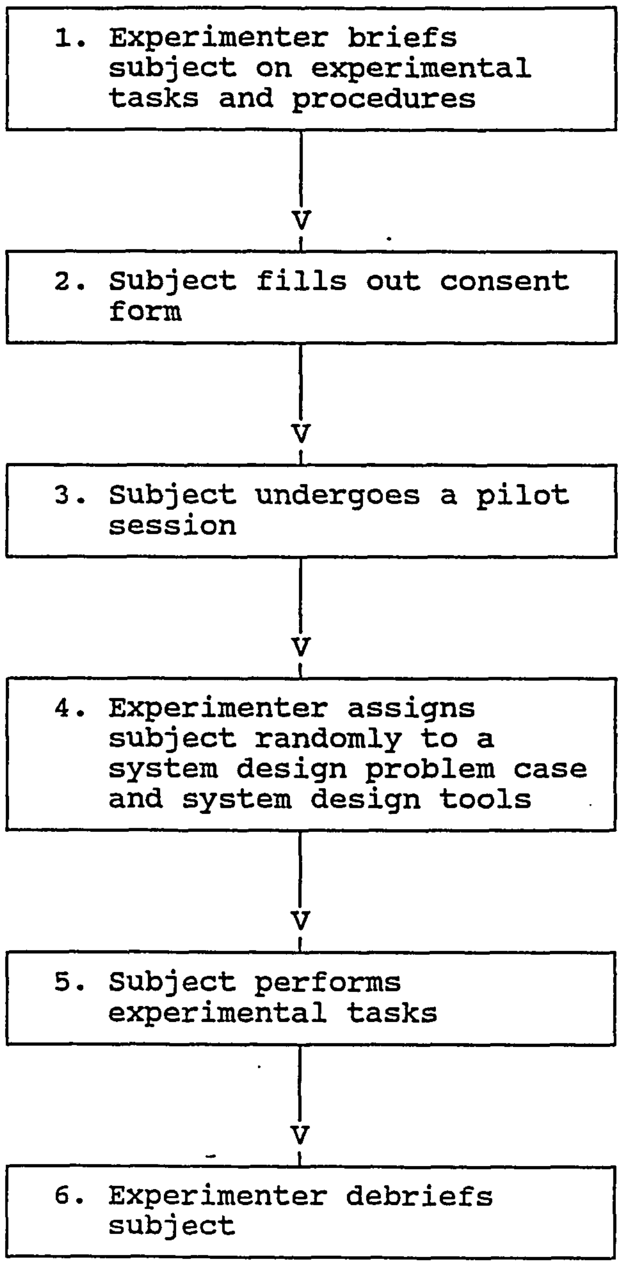

1. Experimenter briefs subject on experimental tasks and procedures
2. Subject fills out consent form
3. Subject undergoes a pilot session
4. Experimenter assigns subject randomly to a system design problem case and system design tools
5. Subject performs experimental tasks
6. Experimenter debriefs subject

Figure 5. Experimental procedures

procedures, and the system design tools to be used in his/her experiment.

Step 4: The subject was randomly assigned to one of the two problem cases (either a less complex problem case or a more complex problem case) and one of the two types of system design tools (either traditional paper-pencil based tool or CASE tool) to be used in the experiment.

Step 5: The subject performed the experimental tasks. The tasks required 2 to 4 hours to complete. The researcher maintained minimum contact with the subject in order to minimize distortion to the experimental data. When the subject decided to stop working on the problem, he or she was requested to turn in all revised data flow diagrams, process descriptions, and data dictionary entries modified and/or produced during the experiment. During the experiment the subject was video taped such that the tape included both the subject as well as an image of working pages (or computer screens) used by the subject. The video-tape provided the "video-protocol" of the way the analyst performed the verification task.

Step 6: Upon the completion of the experiment, the subject was debriefed. The researcher requested the subject to complete a post-experimental questionnaire and explained the nature and contribution of the study.

Experimental Variables

This study has three independent variables and two dependent variables (see Table 10). The three independent variables include the system development tools, system complexity, and system analyst's experience. The two dependent variables in the study are syntactical quality of the design specifications and productivity of the syntactic verification tasks. These variables are described in detail as follows.

System Development Tools Variable and Its Levels

The first independent variable is the type of system development tools used to perform the syntactic verification tasks. This study investigates two types of system development tools: traditional and computer-assisted system development tools. Paper and pencil represent the traditional system development tools. The representative CASE tool as it is well-known and wide use among professional system analysts in real world organizations and educational institutions, represents the computer-assisted tool.

System Complexity Variable and Its Levels

The second independent variable is the level of system complexity. System complexity is defined as the cardinality

Table 10

Experimental Variables and Their Levels

Variable	Level	Description/Measurement
Independent variables:		
1. System development tools	1	Traditional tool (paper and pencil)
	2	Computer-assisted tool (CASE)
2. System complexity	1	Simple system
	2	Complex system
3. System analysts' experience	1	Less experienced system analysts
	2	More experienced system analysts
Dependent variables:		
1. Syntactical Quality Index	$Q = \frac{E_{fc}}{E_s}$	
	where:	
	Q =	Syntactic Quality Index
	E_{fc} =	Number of errors found and correctly changed
	E_s =	Total Number of seeded errors
2. Syntactical Productivity Index	$P = \frac{E_{fc}}{T}$	
	where:	
	P =	Productivity Index
	E_{fc} =	Number of errors found and correctly changed
	T =	Total time on the tasks

(number of instances) of system components (Langefors, 1973; Welke, 1983). This study examines two levels of system complexity: simple and complex systems. Two system specification cases with different levels of complexity were specifically developed for use in this study. The first case, representing the simple system, involved the specification of a billing system for an utility company. The second case, representing the complex system, involved the specification of an inventory control system for a wholesale company.

Billing and inventory control systems were used as the case problems in this study as a majority of the MIS/CIS students (in business school) and professional system analysts are usually familiar with billing and inventory control applications. In the subject screening process, the background questionnaire was used to check each potential subject whether he/she had experience with billing and inventory control systems. Only those subjects who had previous experience with at least one of the two application areas were selected to participate in the experiment.

Based on definitions presented earlier from the work of Langefors and Welke, the two cases used in the experiment were designed to have different levels of system complexity in terms of the number of data flow diagrams, the number of levels in the data flow diagrams, the number of processes, external entities, data storage elements, data flows, and the number of data dictionary entries. Table 11 provides a comparison of the system complexity metrics associated with

Table 11

Comparison of the System Complexity of the Billing System Problem (Simpler) Case and the Inventory Control System Problem (Complex) Case

Dimension	Billing[a] System	Inventory[b] Control System
Number of data flow diagrams	3	10
Number of levels of data flow diagrams	3	4
Number of processes	10	34
Number of external entities	2	4
Number of data storage	2	3
Number of data flows	24	90
Number of data dictionary entries	24	42
Total	68	187

[a] A billing system represents a simple system case.
[b] An inventory control system represents a complex system case.

thesé two cases. As shown in this table, the billing system case (i.e., the simple system) has fewer numbers of data flow diagrams, levels of data flow diagrams, processes, external entities, data storage, and data flows than the inventory control system case (i.e., the complex system). Therefore, the billing system represents a less complex system whereas the inventory control system represents a more complex system.

Appendices C and D provide complete descriptions of the billing system and the inventory control system cases, respectively. In each case, the description of the company background, the experimental tasks and instructions, and the initial system specifications in forms of data flow diagram, process description, and data dictionary are provided on paper as well as on a diskette to be used with the CASE tool.

System Analyst's Experience Variable and Its Levels

The third independent variable represents the level of system analysts' experience. This study examines two levels of the system analysts' experience: less experienced system analysts and more experienced system analysts. Both less and more experienced system analysts have knowledge of structured analysis and design methods and techniques (e.g, data flow diagram, process description and data dictionary); general business functions (e.g., accounting, finance, marketing, and production and inventory management); and the representative

CASE tool. The major difference between these two groups of subjects is that the more experienced system analysts have considerably more "real-world" work experience than the less experienced system analysts. The potential subjects were asked to complete a background questionnaire. The data from this questionnaire was used to determine if each potential subject met the pre-established qualification requirements.

Syntactical Quality Measure

Syntactical quality of the design specifications is one of the two dependent variables. The syntactical quality is defined as the degree to which the design specifications are internally consistent, correct, and syntactically complete (Fraser, Kumar, & Vaishnavi, 1991). The experimental approach adopted in this study is to provide subjects with the initial specifications in which various types of syntactical errors (i.e., internal inconsistency, incorrectness, and incompleteness errors) have been intentionally embedded by the researcher. Table 12 presents a list of the types of errors embedded in the initial design specifications. Appendices E and F show the locations and descriptions of seeded errors in the design specifications of billing system (simple system) and inventory control system (complex system), respectively. In these two appendices, suggested corrections for these seeded errors are also provided. The subjects were asked to diagnose the specification and make all necessary corrections.

Table 12

A List of the Categories and Types of Seeded Errors

Category of Errors	Type of Errors
Incorrectness	1. Incorrect level numbering of a data flow diagram
	2. Incorrect display of file at the level where it is first used
	3. Incorrect display of data flow(s) at the level where it is not first used
	4. Incorrect balance between parent and child data flows into and out of the parent bubbles
	5. Incorrect balance between parent and child data flows into and out of the child diagram
	6. Double-headed arrow in a data flow between processes
	7. Double-headed arrow in a data flow between process and external entity
	8. Incorrect naming of data flows into and out of simple
	9. Incorrect process number
Incompleteness	10. Missing a data flow
	11. Missing data flow name
	12. Missing arrow head in a data flow
	13. Missing data flow definition in data dictionary entry
	14. Missing a data flow diagram
	15. Missing a process
	16. Missing process number
	17. Missing process name
	18. Missing process description
	19. Missing a file
	20. Missing a file name
	21. Missing file definition in Data dictionary
	22. Missing an external entity
	23. Missing an external entity name
	24. Missing process definition in Data Dictionary
Internal Inconsistency	25. Inconsistent process number
	26. Inconsistent process name
	27. Inconsistent file name
	28. Inconsistent external entity name
	29. Inconsistent data flow name

Since the total number of errors seeded in the design specifications was known to the researcher (but not to the subjects), the syntactical quality of final specification can be measured as a percentage of the number of seeded errors found and correctly changed by the subject. In this study, the syntactical quality measure is referred to as "syntactical quality index (Q)." The mathematical equation for computing the syntactic quality index is shown below.

$$Q = \frac{E_{fc}}{E_s} \times 100 \qquad (1)$$

where:

Q	=	Syntactical quality index (as percentage);
E_s	=	Total number of seeded errors; and
E_{fc}	=	Number of seeded errors found and correctly changed.

The range of syntactical quality index (Q) computed by equation (1) must therefore be between 0 and 100 percent. Thus, the higher the Q value, the higher the syntactical quality of the resulting system specification.

Productivity Measure

Productivity of syntactic verification tasks is the second dependent variable in this study. Productivity in its broadest sense is defined as the ratio of output to input.

In the context of this study, the "output" represents the amount of output generated from the syntactic verification tasks performed by the system analyst. The amount of this output was measured by the number of errors in the design specifications found and correctly changed. It should be noted that the errors which the system analyst detects but fails to correct are not considered as output. The "input" in the productivity measure represents the amount of resources spent in generating the output (performing the syntactic verifications tasks). Although various types of resources may be considered as input (e.g., capital, material, personnel, and energy), only the amount of time the system analyst spent on performing the syntactic verification tasks was considered in this study. Therefore, the productivity of syntactic verification tasks is measured as the number of errors found and correctly changed per unit of system analyst time. In this study, this productivity measure is referred to as "syntactical productivity index (P)." The mathematical equation for computing the productivity index is as follows:

$$P = \frac{E_{fc}}{T} \qquad (2)$$

where:

P	=	Syntactic productivity index;
E_{fc}	=	Number of errors found and correctly changed; and
T	=	Total time spent on the tasks.

It should be noted that in equation (2) the higher the P value, the higher the productivity of the syntactic verification tasks.

Data Collection Procedures

In order to measure syntactical quality and productivity, the primary data collected from the experiments include: the number of errors found in the design specifications and correctly changed, and the total time spent in performing the syntactic verification tasks. In addition to the primary experimental data, subjects' attitude toward system development tools and the subjects' task protocol, i.e., the way in which they performed the design specification verification tasks, were also collected.

The following steps were used to collect these data.

Step 1: When the subject started working on the system design problem case, the start time of the experiment was recorded by the experimenter.

Step 2: A video camera was used to visually and audibly record each experimental session in its entirety.

Step 3: When the subject decided to stop working on the problem, the finish time of the experiment was recorded by the experimenter. The revised data flow diagrams, process descriptions, and data dictionary produced by the subject during the experiment were collected for further analysis.

Step 4: Finally, the researcher conducted a post-experimental structured interview with each of the subjects. Appendix I shows an example of the post-experimental structured interview form.

Statistical Analysis Methods

The analyses of the experimental data are organized into four parts. The first part tests the significance of the effects of system development tools, system complexity, and system analysts' experience on syntactical quality of the design specifications and productivity of the syntactic verification tasks. A Multivariate Analysis of Variance (MANOVA) method was used to examine the main and interaction effects of the independent variables (i.e., system development tools, system complexity, and system analysts' experience) on the set of dependent variables (i.e., syntactical quality and productivity). The MANOVA method can identify whether or not the centroids (vectors) of the dependent variables are equal across levels of the independent variables. To facilitate the interpretation of the MANOVA results, a series of Univariate Analysis of Variance (ANOVA) was also performed on each dependent variable.

The second part of the data analysis evaluated the relative performance of traditional and CASE tools with respect to syntactical quality of the specifications and productivity of the syntactic verification tasks. A pair-wise t-test was used to test for differences between the performance of the two system development tools for each combination of system complexity and system analysts' experience levels.

The third part of data analysis examined the impacts of different levels of system complexity and system analysts' experience on the performance of each of the two system development tools with respect to their syntactical quality and productivity. A pair-wise t-test was used to test for differences between the performance of each system development tool as the levels of system complexity and system analysts' experience change. A graphical analysis was also used to assist in interpreting the results from the t-tests.

The fourth part of data analysis involved analyzing the subjects' attitude toward the system development tools and the subjects' task protocol as recorded on the video-tape.

The first three parts of the data analysis are presented in Chapter V. The fourth part is presented in Chapter VI.

Summary

This study uses a controlled laboratory experiment as the research methodology. The experimental design used is a 2^3 factorial design with three independent variables. The independent variables include the type of system development tool (traditional versus CASE), system complexity (simple versus complex systems), and system analysts' experience (less versus more experienced analysts). The dependent variables analyzed in this research include syntactical quality of the design specifications (measured as a percentage of seeded errors found and correctly changed) and productivity of the syntactic verification tasks (measured as the number of seeded errors found and correctly changed per unit of system analyst time).

A sample size of 32 system analysts was used in the experiments. The subjects were classified into two groups: less experienced system analysts and more experienced system analysts. The less experienced system analysts included sixteen students enrolled in advanced system analysis and design courses at Georgia State University and Kennessaw State College. The more experienced system analysts included sixteen professional system analysts who have been working as system analysts with firms in the Atlanta, Sacramento, and San Francisco areas for at least five years.

In the experiments, the subjects were requested to diagnose the provided specifications for internal consistency, correctness, and syntactic completeness; and to make all necessary corrections to the errors found. The subjects were randomly assigned to work on one of the two problem cases (a billing system representing a less complex system case and an inventory control system representing a more complex system case) using one of the two randomly assigned system development tools (traditional versus CASE tools).

The experimental data were analyzed by the MANOVA and ANOVA methods to test the significance of the effects of system development tools, system complexity, system analysts' experience on syntactical quality of the design specifications and productivity of the syntactic verification tasks. The pair-wise t-test was used to evaluate the relative performance of traditional and CASE tools, and to examine the impacts of different levels of system complexity and system analysts' experience on the performance of each of the two tools. Finally, additional analyses were performed to examine the subjects' attitude toward the system development tools and to analyze their video-taped task protocols.

CHAPTER V

EXPERIMENTAL RESULTS

Thirty-two laboratory experiment sessions (one session for each subject) were conducted to collect data for investigating the effects of system development tools, system complexity, and system analysts' experience on the syntactical quality of system specifications and the productivity of the syntactic verification task. Appendix G presents the primary experimental data (i.e., syntactical quality and productivity indices) collected from these experiments. The purpose of this chapter is to present the analyses of the experimental data and discuss the results. The analysis is organized into three parts:

(1) Testing the significance of the effects of system development tools, system complexity, system analysts' experience on syntactical quality of the design specification and productivity of the syntactical verification tasks (research question #1);

(2) Comparing the traditional and CASE tools with respect to their performance in syntactical quality and productivity (research question #2); and

(3) Examining the effects of different levels of system complexity and system analysts' experience on the

syntactical quality and productivity performance of the traditional and CASE tools (research question #3). Finally, this chapter summarizes and discusses the experimental results.

Testing the Significance of the Effects of System Development Tools, System Complexity, and System Analysts' Experience on the Syntactical Quality and Productivity

The significance of the main and interaction effects of the independent variables (i.e., system development tools, system complexity, and system analysts' experience) on the dependent variables (i.e., syntactical quality and productivity) was tested using the MANOVA statistical technique. The MANOVA test can identify if the centroids (vectors) of the dependent variables are equal across all levels of the independent variables. The results of MANOVA are presented in Table 13. Following observations and conclusions can be made from this table.

(1) The MANOVA results show that all main effects (i.e., effects of each of the independent variables, system development tools (T), system complexity (C), and system analysts' experience (E)) are statistically significant at the .0001, .0090, and .0001 levels of significance, respectively. The significance of all main effects (T, C, and E) suggests that a change in any of the independent variables (the type of

Table 13

MANOVA Results

Dependent Variables: Syntactical Quality Index (Q)
Syntactical Productivity Index (P)

Source of Variation	Wilks' Criterion	F Value	Num DF[a]	Den DF[b]	Pr>F[c]
System Development Tool (T)	.217505	41.3723	2	23	.0001*
System Complexity (C)	.664041	5.8182	2	23	.0090*
System Analyst's Experience (E)	.446560	14.2524	2	23	.0001*
T x C	.780443	3.2352	2	23	.0578*
T x E	.499434	11.5260	2	23	.0003*
C x E	.992518	0.0867	2	23	.9173
T x C x E	.955029	0.5415	2	23	.5891

Note: [a] Numerator's degrees of freedom for the F value
[b] Denominator's degrees of freedom for the F value
[c] Significance probability value associated with the F value
* Significance at the .05 level

system development tools, the level of system complexity, or the level of system analyst's experience) can significantly affect the syntactical quality and/or the productivity of syntactic verification tasks.

(2) The MANOVA results further indicate that two-way interactions between system development tools and system complexity (T x C) and between system development tools and system analysts' experience (T x E) are statistically significant at the .0578 and .0001 levels of significance, respectively. The interaction between system complexity and system analysts' experience (C x E), however, is not significant at the .05 level.

The significance of T x C and T x E two-way interactions suggest that the effect of system development tools on the syntactical quality and productivity is contingent upon the level of system complexity and system analysts' experience. These results can be interpreted as follows: the magnitude of changes in the syntactical quality and productivity generated by the two system development tools (traditional and CASE tools) are significantly different from one another when either the level of system complexity or the level of system analysts' experience changes.

(3) For the three-way interaction between the system development tools, system complexity, and system analysts' experience (T x C x E), the MANOVA results indicate that it is not significant at the .05 level.

As explained previously, the MANOVA can identify if centroids (vectors) of the dependent variables are equal across all levels of the independent variables. Inequality of these centroids are confirmed by MANOVA when significant differences in at least one of the dependent variables are detected. MANOVA, however, can not identify whether the differences occur in all dependent variables or only a subset of these variables. Therefore, in order to provide additional insight into effect of the independent variables on each of the dependent variables, two sets of the Univariate Analysis of Variance (ANOVA) were performed. The ANOVA results for the syntactical quality indices and productivity indices are respectively presented below.

ANOVA Results for Syntactical Quality

Table 14 presents the results of the ANOVA analysis for the syntactical quality measure. The following observations are suggested from the examination of this table.

(1) The ANOVA results show that the main effect of the independent variable, system development tools (T) is significant at the .0001 level. The main effect of the independent variable, system complexity (C) is significant at the .0499 level. However, the main effect of the system analysts' experience (E) is found to be significant only at the .1000 level. These results suggest that the syntactical quality is significantly affected by the type of system

Table 14

ANOVA Results on Syntactical Quality

Dependent Variable: Syntactical Quality Index (Q)

Source	DF[a]	Sum of Squares	Mean Square	F Value	Pr>F[b]
Model	7	4395.642187	627.948884	8.90	.0001
Error	24	1693.102500	70.545937		
Corrected Total	31	6088.744687			

R-Square	C.V.[c]	Root MSE[d]	Q Mean
.721929	37.04662	8.399163	22.6718750

Source	DF[a]	Anova SS[e]	F Value	Pr>F[b]
System Development Tool (T)	1	3509.125312	49.74	.0001*
System Complexity (C)	1	300.737812	4.26	.0499*
System Analysts' Experience (E)	1	204.525312	2.90	.1015
T x C	1	286.202813	4.06	.0553*
T x E	1	8.100313	0.11	.7377
C x E	1	9.137813	0.13	.7221
T x C x E	1	77.812812	1.10	.3041

Note: [a] Degrees of freedom
[b] Significance probability value associated with the F value
[c] Coefficient of variation
[d] Square root of the mean square of the error term
[e] Sum of squares
* Significance at the .05 level

development tools and by the level of system complexity, but not by the level of the system analyst's experience.

(2) The ANOVA results indicate that the interaction between system development tools and system complexity (T x C) has a marginal effect on the syntactical quality at the level of significance of .0553. All other two-way interactions (i.e., (T x E) and (C x E)) are not significant at the .05 level. The significance of T x C interaction suggests that the effect of system development tools on the syntactical quality is contingent upon levels of system complexity. This result may be interpreted as follows--the magnitude of changes of syntactical quality resulting from the use of a traditional tool as the level of system complexity changes are significantly different from the syntactical quality resulting from the use of a CASE tool.

(3) Finally, the three-way interaction effect (T x C x E) is not found to be significant at the .05 level.

ANOVA Results for Productivity of the Syntactic Verification Tasks

Table 15 presents the ANOVA results for productivity of the syntactic verification tasks. The following observations and conclusions can be made from this table.

(1) The ANOVA results indicate that all main effects (T, C, and E) are significant at the .0001, .0021, and .0001 levels, respectively. These results suggest that a change in

Table 15

ANOVA Results on Syntactical Productivity

Dependent Variable: Syntactical Productivity Index (P)

Source	DF[a]	Sum of Squares	Mean Square	F Value	Pr>F[b]
Model	7	1326.132297	189.447471	20.38	.0001
Error	24	223.102825	9.295951		
Corrected Total	31	1549.235122			

R-Square	C.V.[c]	Root MSE[d]	P Mean
.855992	44.72821	3.048926	6.81656250

Source	DF[a]	Anova SS[e]	F Value	Pr>F[b]
System Development Tool (T)	1	705.4707031	75.89	.0001*
System Complexity (C)	1	111.0422531	11.95	.0021*
System Analysts' Experience (E)	1	266.7472531	28.69	.0001*
T x C	1	54.1060031	5.82	.0238*
T x E	1	183.5049031	19.74	.0002*
C x E	1	1.2920281	0.14	.7126
T x C x E	1	3.9691531	0.43	.5197

Note: [a] Degrees of freedom
[b] Significance probability value associated with the F value
[c] Coefficient of variation
[d] Square root of the mean square of the error term
[e] Sum of squares
* Significance at the .05 level

any of the independent variables (the type of system development tool, the level of system complexity, or the level of the system analyst's experience) can significantly affect the productivity of the syntactic verification tasks.

(2) The two-way interactions in the ANOVA results indicate two significant interactions effects. The interaction effect of system development tools and system complexity (T x C) is found to be significant at the .0238 level, whereas the interaction effect of system development tools and system analyst experience (T x E) is found to be significant at the .0002 level. The interaction of system complexity and system analyst's experience (C x E), however, is not found to be significant at the .05 level.

The significance of the two way interactions of system development tools and system complexity (T x C) and system development tools and system analyst's experience (T x E) suggests that the magnitude of changes in productivity of the syntactic verification tasks generated by the use of a traditional tool as either level of system complexity or level of system analysts' experience changes are significantly different from that generated by the use of a CASE tool.

(3) The three-way interaction (T x C x E) effect is not significant at the .05 level.

Comparing the Use of Traditional and CASE Tools with Respect to Their Performance in Syntactical Quality and Productivity

The results from the MANOVA and ANOVA analyses suggest that the syntactical quality and productivity measures are significantly affected by the type of system development tools used, and the levels of system complexity and system analysts' experience. Furthermore, the relative differences in performance of the two system development tools (traditional versus CASE tools) seems to be contingent upon the levels of system complexity and system analysts' experience.

Pair-wise t-tests were used to test for differences in the syntactical quality and productivity provided by the two tools under each of the following four conditions:

(1) Less experienced analysts performing the syntactical verification tasks on a simple system.

(2) Less experienced analysts performing the syntactical verification tasks on a complex system.

(3) More experienced analysts performing the syntactical verification tasks on a simple system.

(4) More experienced analysts performing the syntactical verification tasks on a complex system.

Differences in Syntactical Quality
Generated by the Use of Traditional and CASE Tools

Table 16 presents the results of t-test analysis examining the differences in syntactical quality generated by the use of traditional and CASE tools under the above four conditions. The t-test results in Table 16 shows the mean values of the syntactical quality index, the differences in the means of syntactical quality index for the traditional and CASE tools, and the associated level of statistical significance (p) of the difference. In this table, the number in the upper, right-hand corner of each cell is the number assigned to the test condition, and the asterisk (*) indicates that the difference in quality performance between the traditional and CASE tools was significant at the 0.05 level.

The t-test results indicate that differences in the syntactical quality for the two types of system development tools are significant in the following three conditions:

- Less experienced analysts performing the syntactic verification tasks on a simple system (i.e., condition number 1)
- Less experienced analysts performing the syntactic verification tasks on a complex system (i.e., condition number 2), and
- More experienced analysts performing the syntactical verification tasks on a complex system (i.e., condition number 4).

Table 16

Differences in Syntactical Quality Generated by Traditional and CASE Tools (n=4)

Quality Index

		Simple System		Complex System	
	Statistic	CASE	Traditional	CASE	Traditional
			1		2
Less Experienced Analysts	mean	9.075	26.150	11.275	34.075
	mean diff	-17.075		-22.800	
	p	.024*		.001*	
			3		4
More Experienced Analysts	mean	15.175	28.025	13.275	44.325
	mean diff	-12.850		-31.050	
	p	.095		.005*	

Note: Mean diff = mean differences (CASE-Traditional)
p = significance level

The number in the upper, right-hand corner of each cell is the testing condition number.

The asterisk (*) indicates significant differences at .05 level.

In all these three conditions, the traditional tool provides significantly higher syntactical quality than the CASE tool.

In condition number 3 (i.e., more experienced analysts performing the syntactical verification tasks on a simple system), the traditional tool seems to provide a higher syntactical quality than the CASE tool. However, the t-test results indicate that the differences between the two tools is not significant at the .05 level. In this case, the differences, however, are significant only at the .10 level.

Differences in Productivity of the Syntactic Verification Tasks Generated by the Use of Traditional and CASE Tools

Table 17 presents the results of the t-test analysis to examine differences in productivity of the syntactic verification tasks in case of the use of traditional and CASE tools.

At the .05 level of significance, the t-test results indicate that differences in syntactical productivity for the two system development tools are significant under all four testing conditions. The results indicate that the use of the traditional tool provides significantly higher syntactical productivity than the use of the CASE tool under all conditions.

Table 17

Differences in Syntactical Productivity Generated by Traditional and CASE Tools (n=4)

Productivity Index

		Simple System		Complex System	
	Statistic	CASE	Traditional	CASE	Traditional
			1		2
Less Experienced Analysts	mean	0.915	3.620	2.343	8.840
	mean diff	-2.705		-6.497	
	p	.006*		.002*	
			3		4
More Experienced Analysts	mean	2.203	13.078	3.025	20.510
	mean diff	-10.875		-17.485	
	p	.019*		.000*	

Note: mean diff = mean differences (CASE-Traditional)
p = significance level

The number in the upper, right-hand corner of each cell is the testing condition number.

The asterisk (*) indicates significant differences at .05 level.

Examining Effects of System Complexity and System Analysts' Experience on Syntactical Quality and Productivity of the Use of Traditional and CASE Tools

The results from the MANOVA and ANOVA analyses reported previously indicate that the interaction effect between system development tools and system complexity (T X C) was significant for both the syntactical quality and productivity of the syntactic verification tasks. Furthermore, the results suggested that the interaction effect between system development tools and system analysts' experience (T x E) is significant only for the productivity of the syntactic verification tasks and not for the syntactical quality of the resulting specifications. These results, however do not provide any insight toward understanding the nature of these interactions. In order to provide a better understanding of how different levels of system complexity (C) and system analysts' experience (E) affect the syntactical quality and productivity performance of the two system development tools (T), experimental data were analyzed further by a pair-wise t-test at the .05 level of significance.

Effect of System Complexity on Syntactical Quality and Productivity of the Use of Traditional and CASE Tools

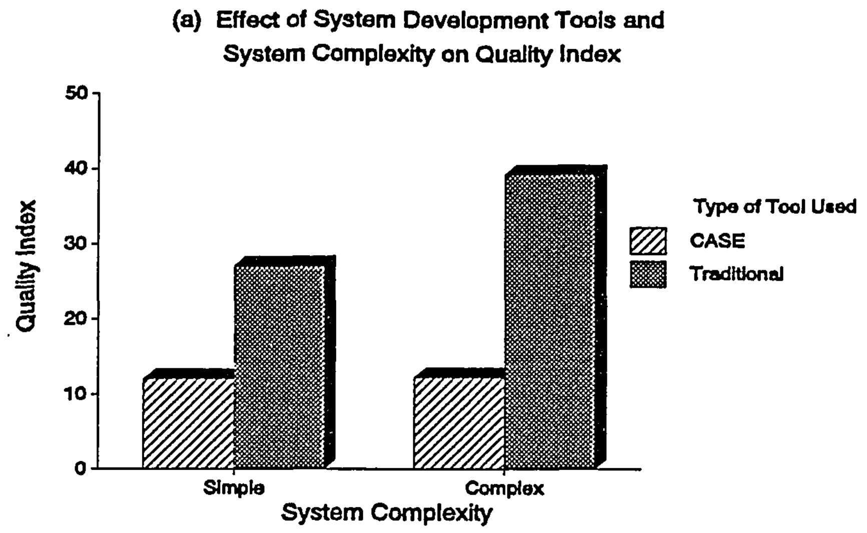
Table 18 presents the results from a pair-wise t-test analysis performed to examine the effects of system complexity

Table 18

Effect of System Complexity on the Syntactical Quality and Productivity Performance of Traditional and CASE tools (n=8)

	System Development Tool			
Factor Level		CASE		Traditional
Effect of System Complexity on Syntactical Quality:				
		Quality Index		
- Simple System		12.125	*	27.088
- Complex System		12.275		39.200
Effect of System Complexity on Syntactical Productivity:				
		Productivity Index		
- Simple System	*	1.559	*	8.349
- Complex System		2.684		14.675

Note: The asterisk (*) indicates significant differences at .05 level of significance.



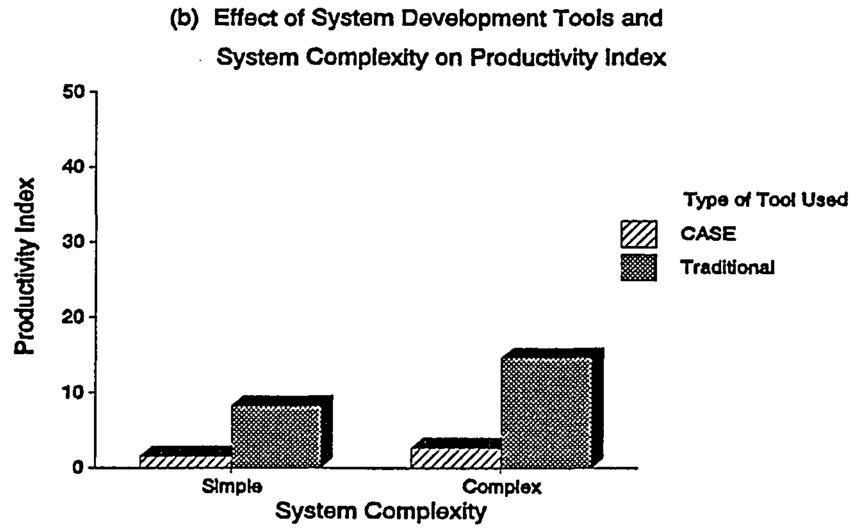


Figure 6. Effect of system development tools and system complexity on the syntactical quality and productivity.

on the performance of the use of traditional and CASE tools. Figure 6 shows a graphical depiction of the results presented in Table 18.

Table 18 presents the mean values of syntactical quality and productivity indices provided by the use of the two system development tools at the two different levels of system complexity (simple versus complex systems). Multiple pair-wise t-tests were performed to test if differences in the syntactical quality and productivity measures between the two levels of system complexity are significance at the .05 level of significance. The results are discussed below.

Effect of System Complexity on Syntactical Quality

The results in Table 18 and Figure 6(a) suggest the following insights into the effect of system complexity on the syntactical quality of the two system development tools.

(1) The interaction between system complexity and system development tools has a significant effect on the syntactical quality.

(2) The use of a traditional tool provides higher syntactical quality than the use of the CASE tool. This implies that levels of system complexity affect the syntactical quality performance of the traditional tool more than the CASE tool.

(3) The syntactical quality generated by a traditional tool improves as the level of system complexity increases. The t-test analysis results presented in Table 19 confirm that this syntactical productivity improvement is significant at the .05 level.

(4) The levels of system complexity have no significant effect on the syntactical quality performance of the CASE tool. The t-test analysis results in Table 18 also confirm that the difference in the syntactical quality generated by a CASE tool as the level of system complexity increases is not significant at the .05 level.

Effect of System Complexity on Productivity of the Syntactic Verification Tasks

The results in Table 18 and Figure 6(b) provide the following with regard to the effect of system complexity on productivity of the syntactic verification tasks of the two system development tools:

(1) The system complexity and system development tools has a significant effect on productivity of the syntactic verification tasks.

(2) The levels of system complexity affect the productivity of the syntactic verification tasks of the use of a traditional tool more than the use of the CASE tool.

(3) Productivity of the syntactic verification tasks generated by using a traditional tool improves as the level of

system complexity increases. The t-test analysis results presented in Table 19 confirm that this syntactical productivity improvement is significant at the .05 level of significance.

(4) The levels of system complexity has some effect on the productivity performance of the CASE tool. Although an improvement in the syntactical productivity resulting from using the CASE tool is quite small, the results of t-test analysis indicate that this improvement is statistically significant at the .05 level of significance.

Effect of System Analysts' Experience on Syntactical Quality and Productivity Performance of the Use of Traditional and CASE Tools

Table 19 presents the results from pair-wise t-tests examining the effects of system analysts' experience on the syntactical quality and productivity of the syntactic verification tasks when using a traditional and the CASE tools. To assist in the interpretation of these results, the results in Table 19 were graphically presented in Figure 7.

Effect of System Analysts' Experience on the Syntactical Quality

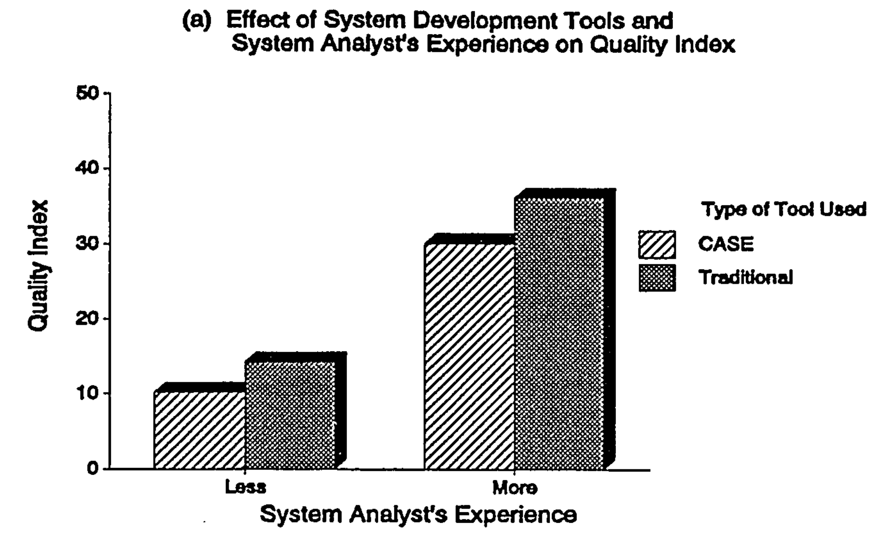
The results in Table 19 and Figure 7(a) provide the following insights into the effect of system analysts'

Table 19

Effect of System Analysts's Experience on the Syntactical Quality and Productivity Performance of Traditional and CASE tools (n=8)

	System Development Tool	
Factor Level	CASE	Traditional
Effect of System Analysts' Experience on Syntactical Quality:		
	Quality Index	
- Less Experienced Analysts	10.175	30.113
- More Experienced Analysts	14.225	36.175
Effect of System Analysts' Experience on Syntactical Productivity:		
	Productivity Index	
- Less Experienced Analysts	1.629	6.230
	*	
- More Experienced Analysts	2.614	16.794

Note: The asterisk (*) indicates significant differences at .05 level of significance.



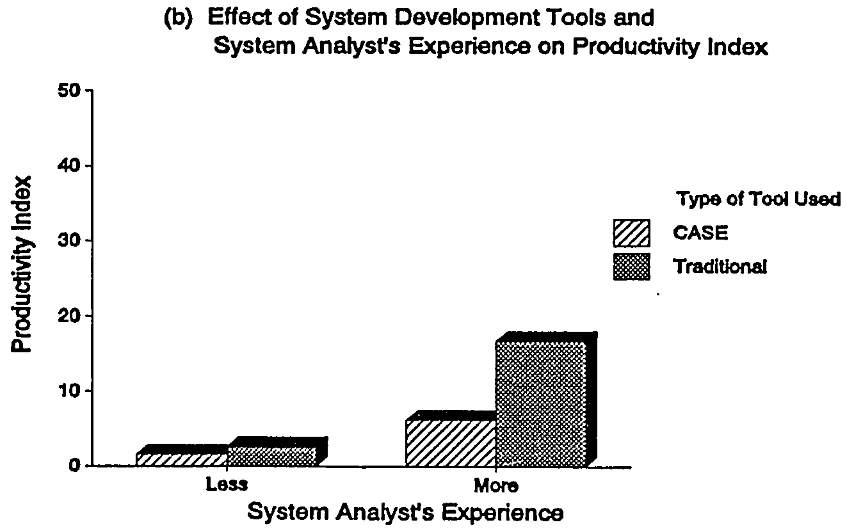


Figure 7. Effect of system development tools and system analyst's experience on the syntactical quality and productivity.

experience on the syntactical quality performance of the two system development tools.

(1) The interaction effects between system analyst's experience and system development tools has no significant effect on the syntactical quality.

(2) The syntactical quality generated by a traditional tool improves as the level of system complexity increases. However, t-test analysis results presented in Table 19 indicate that this syntactical quality improvement when using a traditional tool is not significant at the .05 level of significance.

(3) The syntactical quality generated by the CASE tool improves as the level of system complexity increases. However, the t-test analysis results presented in Table 19 indicate that this syntactical quality improvement when using the CASE tool is not significant at the .05 level of significance.

Effect of System Analysts' Experience on Productivity of the Syntactic Verification Tasks

The results in Table 19 and Figure 7(b) provide the following insights into the effect of system analysts' experience on the productivity performance of the two system development tools.

(1) The interaction between system analysts' experience and system development tools has a significant effect on the productivity of the syntactic verification tasks.

(2) The levels of system analysts' experience affect the productivity performance of a traditional tool more than the CASE tool.

(3) Productivity of the syntactic verification tasks generated by a traditional tool improves as the level of system complexity increases. The t-test analysis results presented in Table 19 confirm that this productivity improvement is significant at the .05 level of significance.

(4) Unlike the use of a traditional tool, the levels of system analysts' experience has no significant effect on the productivity performance of the CASE tool. The t-test analysis results confirm that the difference in productivity of the syntactic verification tasks of the CASE tool is not significant at the .05 level of significance.

Summary and Discussion of Major Findings

The results from the experiments reported previously provide many insights into the research questions that have been stated in Chapter III. The following summarize and discuss the major findings from the experiments as related to the research questions.

In order to investigate the significance of the system development tools, system complexity, and system analysts' experience on the syntactical quality and productivity (research question #1), the experimental data were analyzed by the MANOVA and ANOVA methods. Table 20 summarizes the results of MANOVA and ANOVA. The major findings obtained from these results are summarized and discussed below.

(1) The main effect of system development tools is statistically significant on both the syntactical quality and productivity. This suggests that the use of a traditional tool and the CASE tool provide significant differences in both syntactical quality and productivity. Therefore, a decision to select a system development tool for verifying the specification is critical.

(2) The main effect of system complexity is significant on both syntactical quality and productivity of the syntactic verification tasks. This suggests that different levels of system complexity may significantly change the syntactical quality and productivity of the syntactic verification tasks.

(3) The main effect of system analysts' experience is statistically significant on the syntactical productivity, but not on the syntactical quality. This suggests that a major benefit obtained from using more experienced analysts is an improvement in productivity. The use of more experienced analysts, however, does not substantially improve the syntactical quality.

(4) The interaction between system development tools and system complexity (T x C) has a significant effect on the syntactical quality as well as the syntactical productivity.

Table 20

Summary of the MANOVA and ANOVA Results

Source of Variation	MANOVA Results	ANOVA Results on Syntactic Quality	ANOVA Results on Syntactic Productivity
System Development Tool	Significant	Significant	Significant
System Complexity	Significant	Significant	Significant
Analysts' Experience	Significant		Significant
T*C	Significant	Significant	Significant
T*E	Significant		Significant
C*E			
T*C*E			

Note: Significant at the 0.05 level.
Blank represents insignificant at 0.05 level.
T = System Development Tool
C = System Complexity
E = System Analyst's Experience

This finding suggests that changes in the syntactical quality and productivity are contingent upon the type of system development tools used and the level of system complexity.

(5) The interaction between system development tools and system analysts' experience (T x E) has a significant effect on the syntactical productivity, but not on the syntactical quality.

(6) The interactions between system complexity and system analysts' experience (C x E), and among system development tools, system complexity, and system analysts' experience (T x C x E) have no significant effect on both the syntactical quality and productivity.

A further investigation using pair-wise t-test analysis was performed to determine whether one system development tool always outperforms the other tool in terms of syntactical quality and productivity of the syntactic verification tasks, and if not, what is the relative performance of the two tools under each combination of system complexity and system analysts' experience (research question #2). The results from pair-wise t-test analysis for the syntactical quality and productivity indicate that the use of a traditional tool is better than the use of the CASE tool in terms of the syntactical productivity in all experimental conditions. However, in terms of the syntactical quality the use of a traditional tool is better than the use of the CASE tool in three out of four experimental conditions. The experimental condition number 3 (when more experienced system analysts perform the syntactical verification tasks on a simple system) is the only condition where the syntactical quality levels produced by the two tools are not significantly different. These experimental results do not support the claims by CASE vendors and the expectations by most of the IS practitioners that the use of the CASE tool should lead to an improvement in both system design quality and productivity. On the contrary, the experimental results indicate that the CASE tool performs poorly, in comparison with a traditional tool, in terms of both syntactical quality and productivity. These results motivate us to investigate further into the question of why the CASE tool does not improve the syntactical quality and productivity as expected by most IS practitioners and CASE vendors. This investigation is provided in the next chapter.

Finally, the effects of the different levels of system complexity and system analysts' experience on the performance of traditional and CASE tools in terms of syntactical quality and productivity of the syntactic verification tasks were investigated and summarized as below.

The MANOVA and ANOVA results presented in Table 20 indicate that the interaction between system development tools and system complexity is significant on both syntactical quality and productivity, and the interaction between system development tools and system analysts' experience is significant on syntactical productivity, but not on syntactical quality. However, the MANOVA and ANOVA do not explain exactly how different levels of system complexity and system analysts's experience affect the syntactical quality and productivity performance of traditional and CASE tools. To provide a better understanding into this research question, the experimental data were further analyzed by the pair-wise t-tests and graphical method. The results of these analyses may be summarized as follows.

Effect of System Complexity. For the use of a traditional tool, the level of system complexity has a significant effect on both syntactical quality and productivity. As the level of system complexity increases, the syntactical quality and productivity levels produced by a traditional tool also increase. For the use of the CASE tool, the level of system complexity has a significant effect on only syntactical productivity. The improvement in the level of syntactical productivity level produced by the CASE tool is small as the level of system complexity increases. The level of system complexity has no significant effect on the syntactical quality produced by a CASE tool.

Effect of System Analysts' Experience. For the use of a traditional tool, the level of system analysts' experience has a significant effect on syntactical productivity, but not on syntactical quality. As the level of system analysts' experience increases, the syntactical productivity produced by a traditional tool also increases. For the use of the CASE tool, the level of system analysts' experience has no significant effect on both the syntactical quality and productivity.

CHAPTER VI

EXPLANATION BASED ON DIRECT OBSERVATION AND POST-EXPERIMENTAL INTERVIEW DATA

The experimental results presented in Chapter V suggest that the performance of the system analysts who used the CASE tool was lower than those system analysts who used the traditional paper-pencil based tool. This finding is both contrary to the claims by CASE vendors and inconsistent with the expectations of a majority of CASE users.

The purpose of this chapter is to investigate the reason for these unexpected results. The data collected from direct observation and post-experimental interviews were analyzed to seek answers to these results. It is our conjecture that the patterns of use of the CASE tool and the analyst's attitudes towards the tool would provide insight into these results.

The data analysis in this chapter is organized into three parts. The first part involves the analysis of the system analyst's attitude toward the system development tools. It examines the relative preferences of system analysts for the two development tools and explores the reasons for this preference. In the second part, task-protocols (i.e., the ways system analysts performed syntactic verification tasks) are analyzed in order to determine if there is an association between the types of system development tools, the ways system development tools are utilized, and the levels of syntactical quality and productivity changes. Finally, the third part involves the examination of the use of selected features of the CASE tool and the discussion of the effectiveness of each of these features in assisting system analysts in performing verification tasks.

System Analyst's Attitude Toward System Development Tools

System analyst's attitudinal data were collected through post-experiment interviews. Upon completion of the experimental task, each system analyst (subject) was interviewed regarding their preference for using either tool for the verification tasks. Table 21 presents the results of this interview. The results indicate that 75% of the subjects assigned to the CASE tool in the experiment, and 56% of the subjects assigned to the traditional tool prefer the use of the CASE tool over the traditional tool. Only 25% of subjects assigned to the CASE tool and 44% of subjects assigned to the traditional tool prefer the use of the traditional tool. Overall 67% of the subjects prefer the use of the CASE tool versus only 33% for the use of the traditional tool. These results support the common belief that a majority of system analysts prefer using the CASE tool.

Table 21

A Summary of CASE and Traditional tool Preferences by Users

Type of Tool Used in the Experiment	Number of Subjects	
	Type of Tool Preferred	
	CASE	Traditional
CASE	12 (75%)	4 (25%)
Traditional	9 (56%)	7 (44%)
TOTAL	21 (67%)	11 (33%)

The system analysts were also asked to give reasons to support their preferred choice of system development tools. Common reasons provided by system analysts who chose the CASE tool as system development tool of their choice were as follows.

- CASE tools would assist system analysts in reviewing and revising the previous system design documentation faster and easier than traditional tool.
- CASE tools can be used to update and generate a new system design specification easier and better than traditional tool.
- System design documents developed by CASE tools can be used for further reference and modification.
- CASE tools support and reinforce a structured approach. Therefore, CASE tools should prevent system analysts from making syntactical errors in the system specification.
- CASE tools should assist system analysts in developing the system specification without any errors.
- Data flow diagrams developed and diagnosed by CASE tool are of higher quality than the ones developed by traditional paper-pencil based tools.
- Once the system specification is developed, CASE tools can maintain and regenerate a consistent and good specifications.
- CASE tools provide system analysts with capabilities to perform cross-checking, level balancing, and graphical analysis.

The above data and comment suggest that most of the subjects believe that CASE tool would improve their performance in the syntactic verification of system specifications. On the other hand, results from experiments presented in Chapter V indicate that system analysts who utilized the CASE tool performed poorly with respect to both the syntactical quality and productivity of the syntactic verification task. A relevant question to be asked therefore is: "Despite the apparent preference for the CASE tool, why does the CASE tool degrade the syntactical quality and productivity of system analysts?"

Reasons provided by system analysts who prefer to use the traditional tool (paper and pencil) over the CASE tool may provide some insights into the poor performance of the CASE tool. A list of comments is presented below.

- When using traditional paper-pencil based tools to verify system specification, errors identification and errors correction can be performed concurrently.
- Traditional tools are easy to use and allow system analysts to make numerous copies of similar diagrams.
- Traditional tools may be used to provide a clear idea of the overall scope of system specifications before using the CASE tool.
- Although while using the traditional tools it take longer time to detect all errors and redraw/rewrite specification document, they are easy to use and take less time to correct errors.

- In case of large specifications of 3 to 7 levels or 100 to 300 data flow diagrams, traditional tools provide a complete overview of system specifications but correction of errors at lower levels of specification may be difficult.
- Traditional tools may be useful for a brief sketch of changes or corrections of the specifications.
- Traditional tools are useful for developing a new, specific and unknown application.
- When using traditional tools, system analysts sometime lose track of the latest working version of the specification.

These respondents, therefore, believe that traditional tools are easy to use and take less time to correct errors than the CASE tool. This belief, however, seems to be tempered by the possibility that in complex situations the use of traditional tool may not be adequate. As a result, this belief seems to support most of system analysts' belief that the use of CASE tool may improve their performance in complex situations.

Potential Explanations of Poor Performance Provided by the CASE Tool

The poor performance of the CASE tool may be due to one or more of several potential reasons stated below. The purpose of the following analysis was to determine, as far as possible, which of these potential reasons were likely to contribute to the poor performance while using the CASE tool:

(1) The system analysts who participated in the experiment were not familiar with the applications which they were asked to work with in the experiment.

(2) The system analysts who participated in the experiment did not know how to use the CASE tool.

(3) The CASE tool may have been used in a manner inconsistent with the intended design of the tool.

(4) The CASE tool does not provide features which are easy to use and effective in the verification processes.

At the outset, the first explanation can be ruled out because only those subjects who were familiar with the CASE tool and the billing and inventory control application areas were invited to participate in this study. The subjects also used the inventory control and billing systems tutorial provided by the CASE tool during their training sessions. It can, therefore, be reasonably inferred that the subjects were familiar with these types of applications.

As far as the second explanation is concerned, the subject selection procedure ensured that only those subjects who had been trained on the CASE tool and had some experience with it, participated in the experiment. However, the subject selection procedures did not guarantee that the subjects used the CASE tool as intended by its designer, or were entirely comfortable in using all features of the CASE tool. The answers to these two possibilities mentioned in the last sentence will be investigated while examining explanation (3) and (4) stated above, in detail.

An Investigation of How System Analysts Perform the System Verification tasks Using a Given Tool

A possible explanation of the CASE tool not performing as per expectations may be attributed to the use of the CASE tool in a manner which was not as intended by its designers. It is conjectured that if the CASE tool is not used in a manner consistent with how its designers intended it to be used (as described in the user's manual for the CASE tool), it may not deliver the productivity and quality benefits envisaged by its designers.

In order to investigate this conjecture, the video taped task-protocols of all the subjects were reviewed. The task-protocols were analyzed by preparing a task analysis report which identified, in a step-by-step manner, the activities performed by each analyst. Additionally, comments, if any, made by the analyst while performing these activities were also noted. Appendix H shows examples of the task analysis report for four subjects.

A review of the task-protocols of the subjects suggests that the detailed activities in the verification tasks can be classified into four major categories:

(1) review of the provided system specifications (e.g. problem descriptions, task descriptions, data flow diagrams, data dictionaries, and process descriptions)

(2) identification of possible syntactic errors in the system specification

(3) interpretation of the errors and locating these error on the specification documents or screens, and

(4) correction of the identified errors.

In case of the traditional paper-pencil based tool, the error identification, interpretation and location activities are done by human observer and occur more or less simultaneously. In the situation where the CASE tool is used for the verification task, the CASE tool's "analysis feature" can be used for identifying possible syntactic errors. The interpretation and location of these errors in the specification documents even when using the CASE tool, however, remains essentially a human task.

Although all of the above activities were performed for all treatments in the experiment, differences were observed in the overall patterns in which these activities were performed under different treatments.

Task Pattern within the Traditional Tool Treatment

All subjects using the traditional tool started with a review of the provided system specification and, then, continued to identify, interpret, locate, and correct errors in a concurrent and interleaved manner. That is, a subject would identify and locate the error on the specification and would go on to correct the error before proceeding to the identification and correction of the next error. Thus the four activities of identification, interpretation and location, and correction were all performed in a concurrent manner in this phase. Figure 8 (a) shows the graphical representation of the task pattern within the traditional tool treatment.

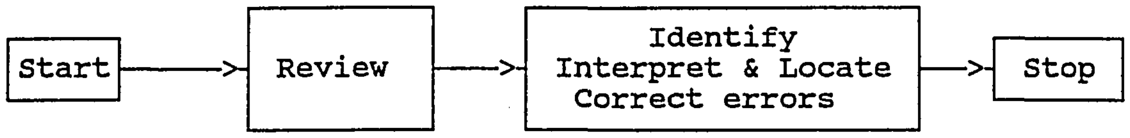
Task Patterns within the CASE Tool Treatment

Three different patterns of activities were observed in the treatments which employed the use of the CASE tool. Figure 8 (b) shows the graphical representation of the task patterns within the CASE tool treatment.

In Pattern A, the subjects started with the review of the system specification on a computer monitor and, then, proceeded to perform a set of concurrent activities of identification, interpretation and location, and correction of errors on a computer monitor. This pattern looks very similar to the traditional pattern identified above.

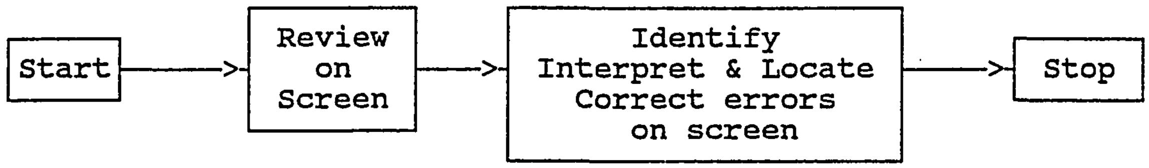
In Pattern B, the subjects started with the review of the system specification on computer monitor and, then performed the set of concurrent activities of identification, interpretation and location, and correction of errors on

(a) Task pattern within the traditional tool treatment:

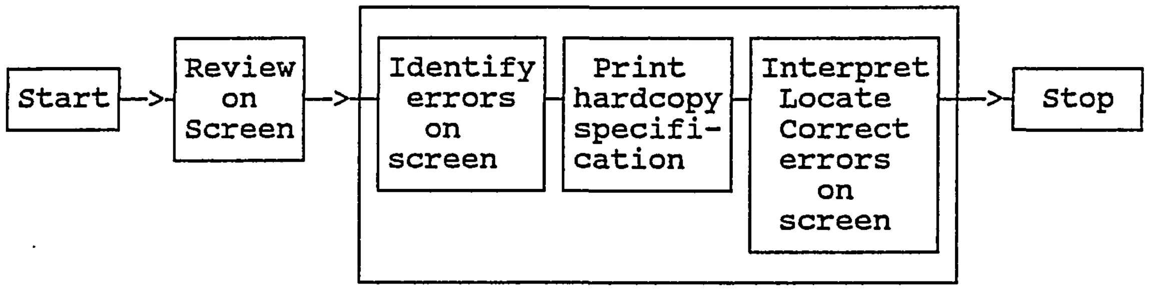


(b) Task patterns within the CASE tool treatment:

Pattern A:



Pattern B:


Start
Review on Screen
Identify errors on screen
Print hardcopy specifi-cation
Interpret Locate Correct errors on screen
Stop

Pattern C:

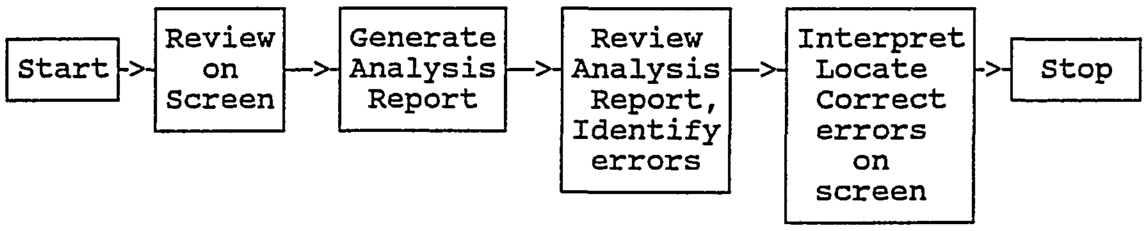

Start
Review on Screen
Generate Analysis Report
Review Analysis Report, Identify errors
Interpret Locate Correct errors on screen
Stop

Figure 8. Task patterns within the traditional tool and the CASE tool treatments

computer monitor in a manner similar to Pattern A. However, the subjects did not locate the errors on the computer monitor of the CASE workstation. Instead, for each error identified, they proceeded to print a out related specification document and located the error on the hardcopy document. Once the error was fully interpreted, the subject went back to the CASE workstation screen to make necessary corrections. Except for the use of the printout, this pattern too is very similar to Pattern A, and the traditional pattern described above.

In Pattern C, as in all other patterns, the subjects started with a review of the system specification on the computer monitor. Next, they proceeded to use the CASE tool's "analysis feature" (see next section) to generate analysis reports for the system specification. The analysis reports generated in this step identify all possible syntactical errors in the specification document (an example of an analysis report is attached in Appendix J). The subjects then, using human thinking and inspection, interpreted the analysis reports and located the errors on the specification displayed on computer screen. Once all the errors were interpreted, located, and diagnosed, the subject went back to the CASE workstation screen to modify the specification in order to correct the errors.

Table 22 summarizes the number of CASE subjects who were classified into each of these patterns within the CASE tool treatment. The table is organized by the level of the system analyst's experience and the level of complexity of the problem. The results in Table 22 are analyzed and discussed as follows.

An Analysis of the Use of Task Patterns within the CASE Tool Treatment

The results from Table 22 indicate that there does not seem to be any differences in the patterns of usage between less experienced and more experienced subjects. It seems as if both sets of the subjects are about evenly divided between pattern A and B (usage patterns which are very similar to the traditional pattern of verification), and pattern C (usage pattern which utilizes the full analysis capabilities of the CASE tool). It seems as if about half of the subjects in the CASE tool treatment are reverting back to the traditional way of identifying errors by inspection and then interpreting and locating errors in the specification. This result is further confirmed by the examination of the use of specific features of the CASE tool discussed in the next subsection. In this section, even those subjects who used the analysis feature express great difficulties in interpreting the analysis report and expressed frustration with the use of this feature.

When Table 22 is examined along the complexity dimension, the results indicate that in a complex situation most subjects are not comfortable working with the computer monitor only. They either print the data flow diagram specification, or work with the printed analysis reports (i.e., Patterns B or C).

Table 22

Task Patterns within the CASE Tool Treatment (n=16)

Level of System Analyst's Experience	Task Pattern	System Complexity		
		Simple	Complex	Total
Less	Pattern A	2	-	2
	Pattern B	-	1	1
	Pattern C	2	3	5
More	Pattern A	2	-	2
	Pattern B	-	2	2
	Pattern C	2	2	4
Total	Pattern A	4	-	16
	Pattern B	-	3	
	Pattern C	4	5	

From this we infer that in a complex situation, if specifications are presented on screen at a time, the subjects do not feel comfortable relating any screen with its proceeding or succeeding screens. In a simple case, however, the number of specification screens is much lower and therefore the subjects were able to keep track of the relationship between the various screens and were able to work directly with the graphic features (i.e., Pattern A) only.

The Relationship between Task Patterns and Syntactic Quality and Productivity of the Verification Task

In order to determine whether the task patterns within the traditional and the CASE treatments have any relationships to the syntactic quality and productivity of the verification process, the following data were gathered from the experiments:

(1) the number of system analysts using each pattern of the syntactic verification activities, and

(2) the levels of syntactic quality and productivity (i.e., mean, minimum, and maximum values) obtained from each pattern group.

Table 23 presents these results. As shown in Figure 8, all sixteen system analysts who were assigned to the traditional tool treatment used the same task pattern. Of the seven system analysts who were assigned to the CASE tool treatment, four of them used Pattern A and three used Pattern

B. The remaining nine system analysts who were assigned to the CASE tool treatment used Pattern C. Table 23 reports the levels of syntactic quality and productivity generated by the subjects in each of these task patterns.

As reported in Table 23, the subjects in the CASE treatment who did not use the analysis feature of the CASE tool (Patterns A and B) had lower productivity and quality performance (i.e., mean, minimum and maximum values) than those subjects who did use the analysis feature (Pattern C). This suggested that inappropriate use of the CASE tool results in lower performance.

However, even those analysts who did use the analysis feature (i.e., Pattern C) still had worse performance than the subjects who used the traditional tool. This is the subject of the following investigation.

An Examination of the Use of Selected Features of the CASE Tool

The CASE tool provides various features that assist the system analysts in developing system specifications. These features include the graphics feature, the data dictionary feature, the screens and reports feature, the documentation feature, the analysis feature, and the housekeeping feature. A brief description of these features is presented in Table 24.

Table 23

Task Analysis Results

Task Patterns	Number of Subjects	Syntactic Quality	Syntactic Productivity
		Mean	Mean
Traditional Tool: Pattern	16	33.1	11.5
CASE Tool:			
Pattern A	4	11.7	1.5
Pattern B	3	7.1	2.3
Pattern C	9	14.1	2.4

The purpose of this section is to examine the effectiveness of the use of the selected features of the CASE tool. The examination focuses mainly on the features needed by the subjects to perform the syntactic verification tasks in the experiment (i.e., the analysis, graphics, and data dictionary features). This examination of the selected features is based on the researcher's direct observations, the study of video task-protocols, and comments by the subjects on these features during the experiment.

Analysis Feature

The analysis feature analyzes the system specification for errors and generates various types of analysis reports. These reports include the graph verification report, the level balancing report, and the extended analysis report.

The graph verification report provides the information about the correctness of data flow diagrams (i.e., omissions, inconsistency and violation of data flow diagram rules). The level balancing report provides information about the completeness and consistency of data flow diagrams from one level to another. The extended analysis report provides information about the completeness, consistency, and redundancy of the data dictionary entries. These entries include data entities, elements, and records specified in the specification.

Table 24

A Brief Overview of the Features of a CASE Tool

Graphics Feature (*)	This feature allows the system analyst to create and update visual representation of the system, its components, and the relationships among them. The analyst can easily modify the graphs and use it to support an iterative systems analysis and design approach.
Data Dictionary Feature (*)	This feature allows the system analyst to define the system and report on the specification data. Data Dictionary provides a central repository for all system information and helps analyst ensure the consistency of specification
Screens & Reports Feature	This feature allows the system analyst to develop working models of system's input screens and output reports.
Documentation Feature	This feature allows the system analyst to produce hardcopy output of every aspect of the system, organized according to an outline specified.
Analysis Feature (*)	This feature assists the system analyst in ensuring the consistency and accuracy of data by providing reports for examining data and verifying its adherence to standard techniques.
Housekeeping Feature	This feature provides functions for establishing and maintaining projects, users, and hardware.

Note: (*) = The feature used by the subjects in performing various task related activities.

Only 9 out of 16 subjects who were assigned to the CASE tool treatment used the analysis feature to perform the syntactic verification tasks. They spent only a few minutes preparing the request for generating the analysis reports, but took 10 to 25 minutes to print 20 to 80 pages of the analysis reports. They did not show any frustration or make any complaints up to this point in the experiment. This suggests that using the analysis feature makes it easy to generate the analysis reports.

When the subjects started examining the analysis reports, most of them had difficulties with the interpretation of the results presented in these reports. The time used by the subjects to interpret the reports and locate errors on the screen was very high. Precise estimates of these times, however, are not possible as the error correction activities were interleaved with this task.

The subjects comments and complaints about the analysis reports are summarized as follows.

- The analysis reports were unreasonably long and difficult to interpret.
- These reports did not provide the critical information required for identification and location of errors on the specifications.
- The reports did not provide any alternative solutions or recommendation for correction of the identified errors.

- The system analysts suggest that these reports should provide a layout of the scope of the real problem and a list of all users' names and their requirement.
- The reports should provide information that assist the system analysts in identifying what to do and how to do it including the backup of specification before and after changing or correcting errors.
- The analysis feature should allow the system analyst to perform the analysis without following a set of sequential steps because a majority of system analysts do not use a "Top down" approach.
- The system analysts may generate and examine all reports. However, they sometime do not know the meaning of the results.
- The reports do not tell the system analysts what they want to know and make them feel very uncomfortable.
- The system analysts who determine not to continue using analysis feature comment that they can not understand what is going on inside once they have changed or corrected the specifications.

The above comments and the inordinate time requirements for using analysis feature suggest that the analysis feature is not easy to use and does not provide easy to understand information needed in the identification, interpretation and location, and correction of errors in the system specifications. As a result, the excessive time spent in requesting, printing and interpreting the analysis reports reduce the productivity of the use of the CASE tool and the difficulties of interpreting the analysis reports may reduce the quality of the system specification.

Graphics Feature

The graphics feature provides the CASE subjects with a capability to delete, add, or modify data flow diagrams. This feature is useful in correcting errors in the design specifications. In the experiment, all of the CASE subjects used the graphics feature for correcting the identified errors. In case of the simple specification, the CASE subjects did not show any frustration with the graphics feature. However, in case of the large and complex specification, the CASE subjects did not feel comfortable and had difficulties with the graphics feature when correcting errors displayed on the data flow diagrams. The subjects provided several reasons for their frustration and difficulty. Representative comments from the subjects are:

- In a complex specification, the graphics screen is crowded with a cluster of entities, processes, data flows, and data stores.
- It is difficult to correct errors on crowded screens.
- When one error correction is made on a displayed data flow diagram, the graphics feature would automatically redraw the diagram which, in turns, make it difficult for the system analysts to keep up with the changes.

- The graphics feature did not use color to identify and locate errors.
- Although the graphics feature supports the iterative errors correction activities, it took the analysts too much time and too many steps to retrieve one data flow diagram and make correction to just one error.
- The graphics feature should provide the analysts with hot keys to move from one point to any other points in the specification without having to go through several menus.
- Although the graphics feature is useful in correcting errors, it does not help the system analysts to correct errors as quickly as they want.

These comments suggest that graphics feature is not easy to use for locating and correcting the errors in complex specifications.

Data Dictionary Feature

The data dictionary feature may be considered as a core feature of CASE tool for developing the system specification. This feature assists the system analysts in defining all the elements of the structured specification, i.e., entities, processes, data flows, and data storage. In the experiment, only 4 out of 16 CASE subjects used data dictionary feature. The followings are plausible reasons, summarized from the subjects' comments, of why a majority of subjects using the CASE tool did not use the data dictionary feature:

- The data dictionary feature is not easy to use.
- It does not provide meaningful information for identifying and correcting the errors.
- Its reports are too long and difficult to interpret.

These reasons suggest that the data dictionary feature is also difficult to use.

In summary, the results from the examination of each selected features of the CASE tool as presented above suggest that these features are difficult to use and time consuming. These difficulties in using these features could be the reason why productivity was low, and why system analysts using the CASE tool did not discover and correct all errors and, therefore, quality was low.

Summary

This chapter investigates the reasons for the result of poor performance of the CASE tool presented in Chapter V. The investigation is based mainly on direct observation and post-experimental interview data. The results from analysis of the system analyst's attitudes toward the CASE tool, the task-protocols, and the examination of the use of selected features of the CASE tool indicate that the poor performance of the CASE tool seems to be due to the inappropriate use of the CASE tool and the limitations of its analysis, graphics, and data dictionary features.

CHAPTER VII

CONCLUSIONS, IMPLICATIONS, LIMITATIONS AND DIRECTIONS FOR FURTHER RESEARCH

The purpose of this chapter is to present conclusions, summary of the major findings, implications, limitations and future research directions from this research.

Conclusions

The objective of this research was to investigate the effect of the use of CASE tools on syntactical quality of system specification and productivity of the syntactic verification tasks under different levels of system analysts' experience and system complexity.

A controlled laboratory experiment was conducted to achieve the research objective. A multi-variate analysis of variance (MANOVA), an analysis of variance (ANOVA), and a pair-wise t test methods were used to quantitatively analyze experimental data. A protocol analysis of direct observation and video tape was used to qualitatively explain results from analysis of the experimental data.

A summary of the major findings is presented in the following section.

Summary of the Major Findings

The major findings from the controlled laboratory experiment may be summarized as follows:

(1) The use of CASE tool provide lower quality and productivity performances than the use of traditional paper-pencil based tool.

(2) System complexity and system analyst's experience do not seem to affect quality and productivity performances of the use of CASE tool.

(3) If CASE tool is used as intended by its designer, it provides better quality and productivity than when it is used in the same manner as traditional tool. However, the use of CASE tool as intended still provides lower quality and productivity performances than traditional tool.

(4) The problem of poor performance of CASE tool seems to lie in the way each feature of CASE tool is used (e.g., difficult to use and connect information).

Implications of the Research

The results of this research may have implications for three groups of professionals in the management of information systems area: designers of CASE tools, adopters and implementers of CASE tools, and MIS researchers.

Designers of CASE Tools

The findings from this study suggest that when designing features and functions of CASE tools, designers of CASE tools should take into account the way CASE users actually analyze (i.e., verify and correct) system specifications. The implications from the findings are presented as follows.

Possibility of integrating hyper-media technology into the user-interfaces of CASE tools. The results presented in chapter VI suggested that one of the major problems with the use of CASE tools was due to navigational problems in connecting information on one system representation screen (e.g., a particular data flow diagram) to information on other system representation screens (such as higher or lower data flow diagrams, data dictionary, or process descriptions). These navigational problems can be ameliorated by the use of hyper-media interface which would give the analyst the capability of retrieving the details behind any aspect of a representation directly by connecting it to other representations. The use of such interface would make it much easier for the analyst to detect, analyze, and correct errors using the CASE tools.

In addition, the development of multi-media based tutorials and training sessions is also suggested to be incorporated into the CASE tools. In chapter VI, it is indicated that a number of system analysts did not use the CASE tool in a manner which is intended by its designers.

The use of multi-media based CASE tutorials and training sessions will provide consistent and continuous training supports that correspond to individual system analysts learning needs.

Adopters and Implementers of CASE Tools

The findings from this study suggest that when acquiring the CASE tools, adopters and implementers of CASE tools should perform a careful evaluation of CASE tool features with respect to their compatibility with how system analysts actually analyze the system design specifications. They also need to be trained on how to correctly use CASE tool. In the case of experienced system analysts, the adopters need to make sure that they will not carry over the habits acquired while they were doing traditional paper-pencil based analysis.

MIS Researchers

Currently, MIS researchers do not know how system analysts perform their tasks or how they interface with automated support systems such as CASE tools (or traditional tools). They need to study and describe how system analysts perform all of their tasks. Possibly, a detailed Protocol study of system analysts' behavior and problem solving processes is needed before CASE designers or implementer take assessment of CASE tools.

Limitations of the Research

Several limitations of the study are addressed in this section. First, only two types of design tools (i.e., traditional and CASE tools) were tested. One common CASE product was selected and used as the representative CASE tool in this study. There are other CASE tools/products available in the market. This study does not attempt to evaluate all CASE tools/products. It seeks to specifically compare effectiveness between the use of traditional and the CASE tool.

The study is also limited in terms of system development tasks investigated in the research. Within a system development life cycle, there are many tasks to be performed at each of various phases in the life cycle (see Davis & Olson, 1985). This study focuses on investigating effectiveness of the use of traditional tool versus the CASE tool in performing system specification verification tasks. Research investigations similar to this study should be carried out to examine the effectiveness of the CASE tools in other phases in system development life cycle. Generalizing the results from this control laboratory experiment research to other system development tasks or phases may not be proper.

Another limitation of the study is concerned with the type of subjects used. The sample from schools, companies in Atlanta, Sacramento metropolitan and nearby areas may not be characteristic of system analysts throughout the United

States. A larger sample of system analysts with qualification and experience in system analysis and design will improve validity of this study. Furthermore, if we can differentiate between more experienced subjects who have only methodology experience versus CASE tool experience, we will be able to distinguish the effect from these two different types of experience.

A further limitation of the study lies in the domain of the study itself. There is a larger set of variables which may potentially affect productivity and quality of system development. This study is considered as an initial effort of a long-term research project in system development quality and productivity area. Although this research domain is limited, it will provide a considerable contribution to both MIS and CASE literature.

Directions for Further Research

This research represents an experimental investigation of Computer-Aided Software Engineering technology and its effects on quality of the system design specifications and productivity of the system design verification process under different levels of system analyst's experience and system complexity. This study can be extended and replicated in several directions.

(1) Extend the investigation to understand each individual system analyst's behavior and his/her problem solving process.

(2) Extend the investigation to understand a team of system analysts' behavior and their problem solving process.

(3) Perform the investigation using different CASE products and components.

(4) Perform the investigation using more levels of system complexity in order to identify which levels of the system complexity at which CASE tools may provide advantages over traditional tool.

(5) Perform a similar investigation within a specific company where incentive for participation and completion of verification of complex system using CASE tools is offered.

(6) Perform a similar investigation using different applications to identify which types of applications CASE tools will provide quality and productivity advantages.

APPENDIX A

CONSENT FORM

I understand that it is the best interest of scientific inquiry not to discuss with my fellow students or colleagues any aspect of the experiment in which I am participating. I fully realize that such discussion may lead to possible distortions of data and may in effect cause the entire experiment to be abandoned. I agree to keep the experiment confidential.

______________________	______________________
Signature	Date

APPENDIX B

SUBJECT BACKGROUND QUESTIONNAIRE

Please fill in your name, address, and telephone number:

Name ____________________
Company Name ____________________
Dept/Mail Stop ____________________
Address ____________________
____

Telephone (_____)______________

ORGANIZATION:

1. What is the primary end-product of your company? (Check one)

1( ) Manufacturing	5( ) Financial Services
2( ) Consulting	6( ) Computer/EDP Services
3( ) Education	7( ) Government Agency
4( ) Utilities	8( ) Other________________

2. How many people are employed in your company?

1( ) Over 1000	4( ) 50-100
2( ) 500-1000	5( ) Under 50
3( ) 100-500	

3. How many people are employed in your department?

1( ) Over 100	4( ) 10-25
2( ) 50-100	5( ) Under 10
3( ) 25-50	

4. How many people in your department use Excelerator?

1( ) Over 20	4( ) 1-5
2( ) 10-20	5( ) None
3( ) 5-10	

5. If your department use Excelerator, what is the average size of projects on which Excelerator has been used?

1( ) Over 36 man-months	4( ) 6-12 man-months
2( ) 24-36 man-months	5( ) Less than 6 man-month
3( ) 12-24 man-months	6( ) Other______________

6. If you use Excelerator, how long have you been using Excelerator?

1( ) Over 36 man-months	4( ) 6-12 man-months
2( ) 24-36 man-months	5( ) Less than 6 man-month
3( ) 12-24 man-months	6( ) Other_______________

7. How many systems were developed by you or by you and you project team using Excelerator during the last three year?

1( ) Over 4 systems	4( ) 2 systems
2( ) 4 systems	5( ) 1 systems
3( ) 3 systems	

8. The systems developed by you or by you and your project team fall into which of the following categories? (Check all that apply)

1( ) Billing Systems	4( ) Financial Management
2( ) Inventory Systems	5( ) Database Application
3( ) Communications	6( ) Others______________

JOB ROLE:

9. What is your job title?________________________________

10. What is your job role in relation to projects using Excelerator? (check one)

1( ) Manager of department where Excelerator is used
2( ) Manager of project team using Excelerator
3( ) Member of project team using Excelerator
4( ) Librarian using Excelerator on behalf of team
5( ) User of the systems developed by project team
6( ) Not related to project using Excelerator

11. To what percentage did each of the following activities constitute a part of your responsibilities in the last six months? (Total should be less than or equal to 100%)

Strategic planning	_____________%
Feasibility	_____________%
Requirement	_____________%
Analysis	_____________%
Design	_____________%
Coding	_____________%
Testing	_____________%
Maintenance	_____________%

12. How many years have you performed system analysis and design tasks?

1( ) Over 20 years	4( ) 2-4 years
2( ) 10-20 years	5( ) Less than 2 years
3( ) 5-10 years	

13. How many systems have you developed and completed during the last five years?

1( ) Over 15 systems	4( ) 3-5 systems
2( ) 10-15 systems	5( ) Less than 3 systems
3( ) 5-10 systems	

14. How many systems have you developed using structured System Analysis and Design Technique (Data Flow Diagram Technique) during the last five years?

1( ) Over 15 systems	4( ) 3-5 systems
2( ) 10-15 systems	5( ) Less than 3 systems
3( ) 5-10 systems	

15. How many years have you performed programming tasks?

1( ) Over 20 years	4( ) 2-4 years
2( ) 10-20 years	5( ) Less than 2 years
3( ) 5-10 years	

16. How many programs have you written and completed during the last five years?

1( ) Over 30 programs	4( ) 5-10 programs
2( ) 20-30 programs	5( ) Less than 5 programs
3( ) 10-20 programs	

APPENDIX C

SIMPLE SYSTEM DESIGN PROBLEM CASE

Problem Description

The Watts Electric company is a large company that sells electricity to local customers in a small city. The company wants to develop a new computer-based billing systems to improve the current customer billing process. The input to the system will consist of meter number and the most recent reading from the meter reader expressed as a six digit number. The output of the system is a customer monthly statement. Once the data are input, they will be processed by a computer program that searches a file to match the input meter number with one stored in the file. This will make it possible to access the record corresponding to the customer who has been assigned the meter. The number is the key for the customer record. Other data contained in the record include customer name and address, last meter reading, billing code, and any unpaid balances from previous billing periods.

When the customer record is accessed, it is used to calculate the next bill and prepare a statement that can be mailed to the customer. The amount of electricity used is calculated by subtracting the previous meter reading from the new reading. The amount of electricity used is then multiplied by the appropriated unit charge, which is determined by using the customer's billing code and matching it in a file contains all unit charges. Thus, if the customer's billing code is "A" the code file is checked to determine the unit charge that corresponds to the code of "A".

Once the amount of the current bill has been calculated, a statement is output from the computer system. The statement contains the customer's name and address, the date of the statement, the amount of electricity used, the unit charge, the beginning and ending meter readings, the amount of the bill for the period, the last date to pay the bill, and the dated of the next meter reading. The bill shows franchise charges and utility taxes, along with any balance the customer owes on the preceding month's bill. Finally, the statement gives a grand total of charges and taxes.

Tasks Description

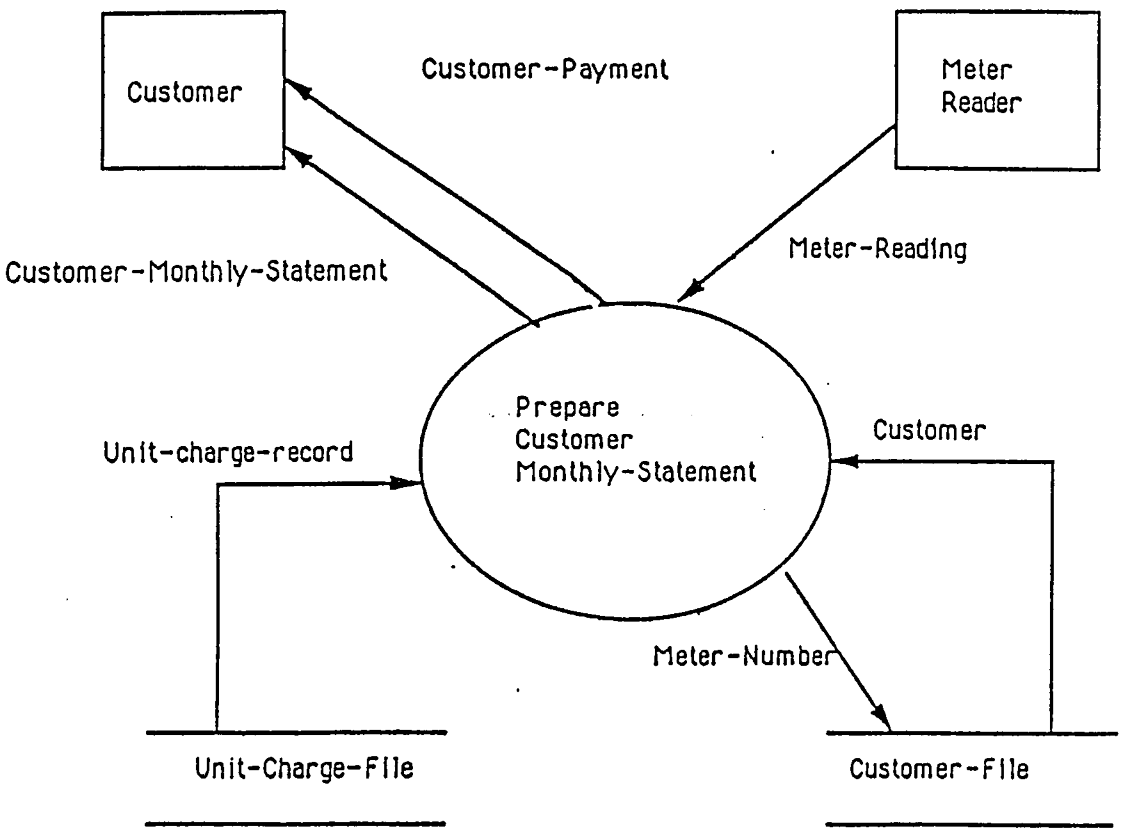
The new system specifications were developed by an analyst who was re-assigned to another project. You have been assigned to this project as a system analyst. Before you start additional work, you need to make sure that the system specifications are correct, complete, and consistent. Your tasks include:

1) To verify the system specification for correctness, completeness, and consistency.

2) To modify and correct the system specification so that the above errors are corrected. When you find an error, you can let the interviewer know the error found. Please also explain to the interviewer any corrections to the system specification you are making to correct those errors.

DATA FLOW DIAGRAMS

CONTEXT DIAGRAM OF BILLING SYSTEMS



Data Flow Diagram Level 1

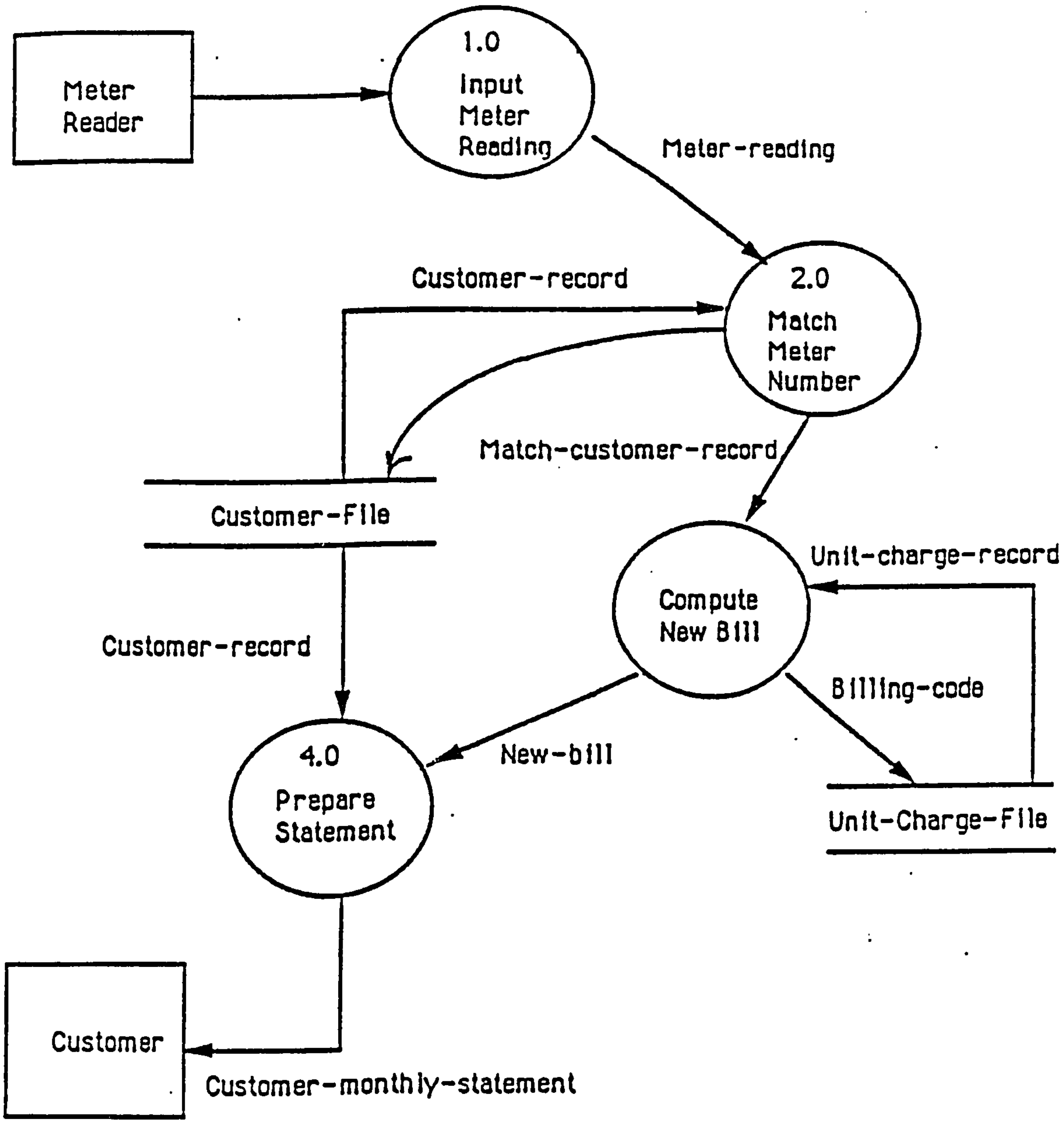
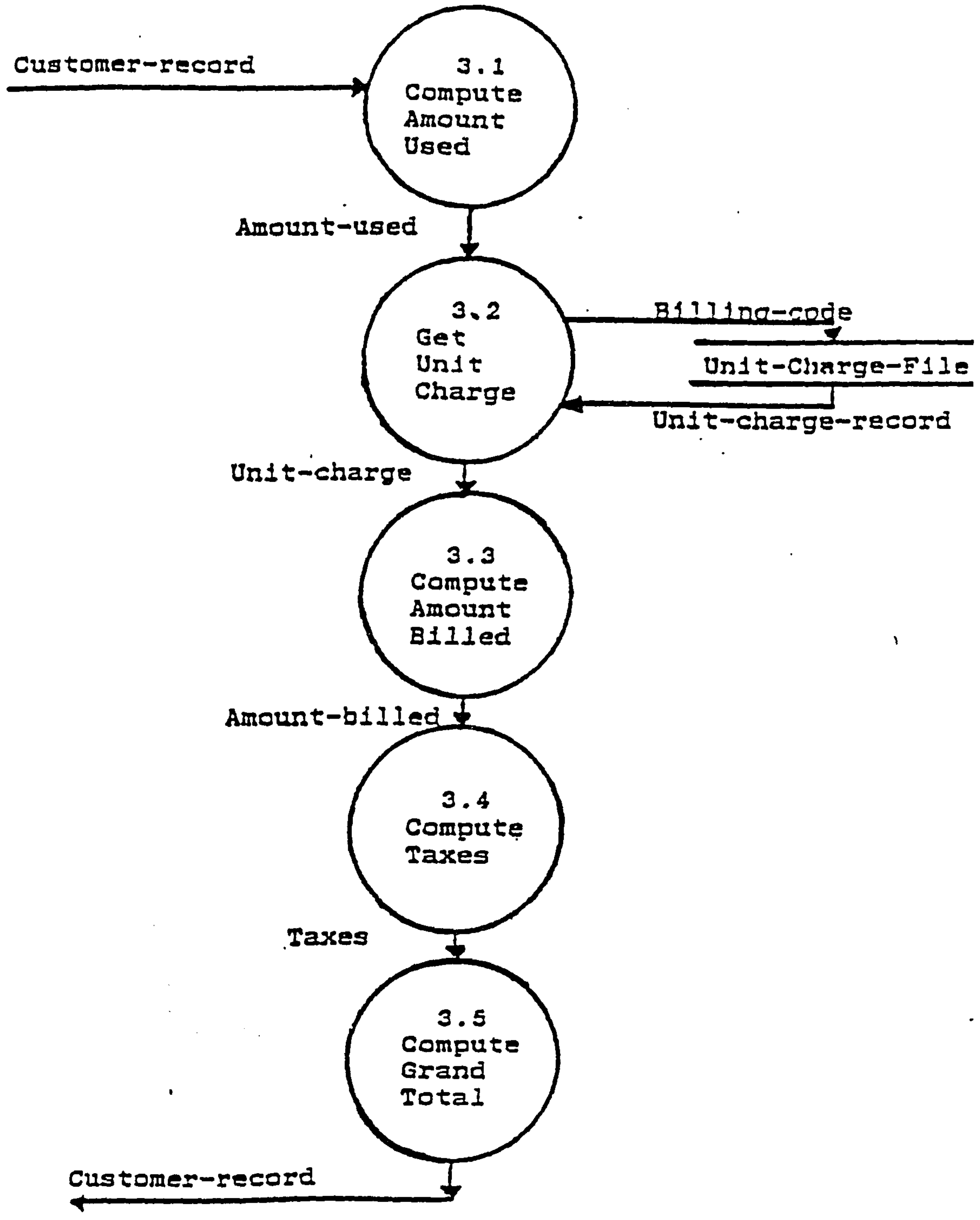


DIAGRAM 3


Customer-record
3.1
Compute
Amount
Used
Amount-used
3.2
Get
Unit
Charge
Billing-code
Unit-Charge-File
Unit-charge-record
Unit-charge
3.3
Compute
Amount
Billed
Amount-billed
3.4
Compute
Taxes
Taxes
3.5
Compute
Grand
Total
Customer-record

PROCESS DESCRIPTIONS

PROCESS DESCRIPTION FOR SIMPLER SYSTEM PROBLEM

PROCESS DESCRIPTION

PROCESS NUMBER:	1.0
GRAPH NAME:	DFD-LEVEL1
PROCESS LABEL:	INPUT-METER-READING

DESCRIPTION: The input data from the meter reader consists of meter-number and the most recent reading from the meter reader expressed as a six digit number is received and forwarded to the "Match-Meter-Number" process.

PROCESS DESCRIPTION

PROCESS NUMBER:	2.0
GRAPH NAME:	DFD-LEVEL1
PROCESS LABEL:	MATCH-METER-NUMBER

DESCRIPTION: The input "Meter-number" will be used to search a "Customer-File" until it finds its matched "Customer-record" stored in the file. The "Meter-number" is the key for the customer record. The match "Customer-record" is forward to the "Compute-New-Bill" process.

PROCESS DESCRIPTION

PROCESS NUMBER:	3.0
GRAPH NAME:	DFD-LEVEL1
PROCESS LABEL:	COMPUTE-NEW-BILL

EXPLOSION DFD: DIAGRAM 3

DESCRIPTION: The match "Customer-record" is received. The information inside the record is used to calculate the new bill. The "Customer-record" with the new bill information is forward to the "Prepare-statement" process.

PROCESS DESCRIPTION

PROCESS NUMBER:	3.1
GRAPH NAME:	DIAGRAM 3
PROCESS LABEL:	COMPUTE-AMOUNT-USED

DESCRIPTION: The "Amount-used" or amount of electricity used is computed by subtracting the "Beginning-meter-reading" from the "Ending-meter-reading".

PROCESS DESCRIPTION

PROCESS NUMBER:	3.2
GRAPH NAME:	DIAGRAM 3
PROCESS LABEL:	GET-UNIT-CHARGE

DESCRIPTION: The "Unit-charge" is retrieved from the "Unit-charge-record" in the "Unit-Charge-File" by matching the "Billing-code" from the "Customer-record" with the "Billing-code" in the "Unit-charge-record" in the "Unit-Charge-File".

PROCESS DESCRIPTION

PROCESS NUMBER:	3.3
GRAPH NAME:	DIAGRAM 3
PROCESS LABEL:	COMPUTE-AMOUNT-BILLED

DESCRIPTION: The "Amount-billed" is calculated by multiplying the "Amount-used" by "Unit-charge".

PROCESS DESCRIPTION

PROCESS NUMBER:	3.4
GRAPH NAME:	DIAGRAM 3
PROCESS LABEL:	COMPUTE-TAXES

DESCRIPTION: The "Taxes" is computed by multiplying the "Amounted-billed" by the "Utility-tax".

PROCESS DESCRIPTION

PROCESS NUMBER:	3.5
GRAPH NAME:	DIAGRAM 3
PROCESS LABEL:	COMPUTE-GRAND-TOTAL

DESCRIPTION: The "Grand-total-charge" is equal to "Unpaid-balance" plus "Amounted-billed" plus "Taxes" and plus Franchise-charge.

PROCESS DESCRIPTION

PROCESS NUMBER:	4.0
GRAPH NAME:	DFD-LEVEL1
PROCESS LABEL:	PREPARE-STATEMENT

DESCRIPTION: The "Customer-monthly-statement" is prepared and printed. The statement contains the "Customer-record" forwarded from process 3.0. When finish, each "Customer-record" is saved in the "Customer-File".

DATA DICTIONARY

DATA DICTIONARY FOR SIMPLER SYSTEM PROBLEM

```
Amount-billed    = Amount-used + Unit-charge
Amount-used
Beginning-meter-reading
Billing-code
Customer-address= Street + City + State + Zip-code
Customer-File    = {Customer-record}
Customer-monthly-statement =  Customer-record
Customer-name    = First-name + Last-name
Customer-record = Meter-number + Customer-name
                 + Customer-address
                 + Date + Amount-used + Unit-charge
                 + Beginning-meter-reading +
Ending-meter-reading
                 + Last-meter-reading + Billing-code
                 + Unpaid-balances + Amount-billed
                 + Last-date-to-pay + Date-next-meter-reading
                 + Utility-tax + Grand-total-charge
                 + Franchise-charge
Date
Date-next-meter-reading
Ending-meter-reading
Franchise-charge
Grand-total-charge= Unpaid-balance + Amount-billed + Taxes
Last-date-to-pay
Last-meter-reading
Meter-number
Meter-reading    = Meter-number + Recent-reading
Taxes            = Amount-billed + Utility-tax
Unit-charge
Unit-charge-File = {Unit-charge-record}
Unit-charge-record= Billing-code + Amount-used + Unit-charge
Unpaid-balances
Utility Tax
```

APPENDIX D

COMPLEX SYSTEM DESIGN PROBLEM CASE

Problem Description

Company A is a large wholesale company that purchases and sells measuring equipment. Its annual revenues at the close of third quarter in 1989 were $205 million. Company revenues are growing at annual rate of 25 percent. Company A's success stems from its ability to monitor and meet customer demand. The items sold to customer are at a large volume and discount rate. The fast delivery service of items from warehouse to customers is the key to business success. Recently, company A decided to improve its delivery service by developing a new computer-based inventory control system for its warehouse. The new system will be used to: 1) keep track of the level of inventory of each items in stock, on purchasing process, and on receiving process, 2) process customer orders forwarded from marketing department, and 3) prepare shipping schedule, ABC analysis report, and cycle counting report to management.

Tasks Description

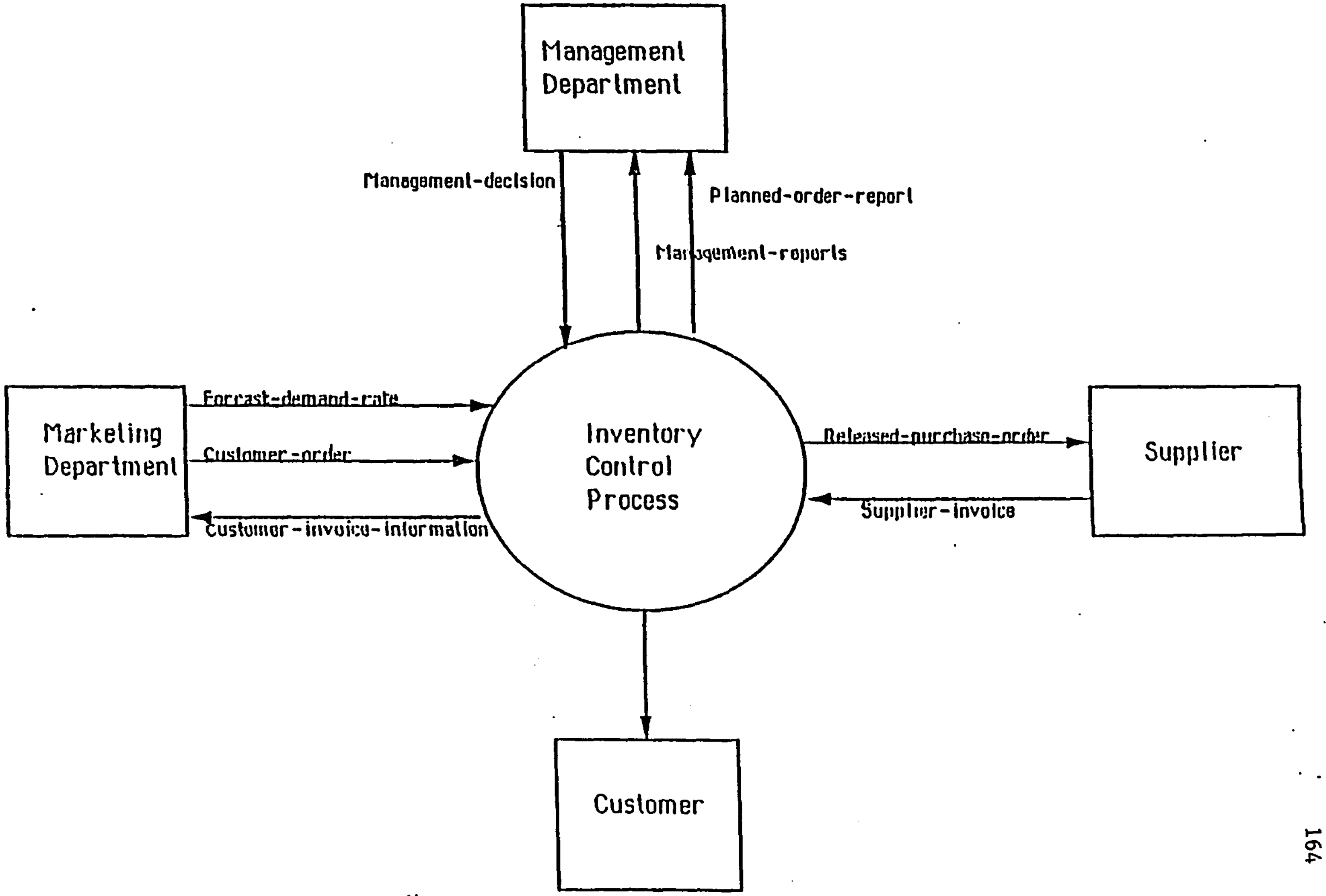
The new system specifications were developed by an analyst who was re-assigned to another project. You have been assigned to this project as a system analyst. Before you start additional work, you need to make sure that the system specifications are correct, complete, and consistent. Your tasks include:

1) To verify the system specification for correctness, completeness, and consistency.

2) To modify and correct the system specification so that the above errors are corrected. When you find an error, you can let the interviewer know the error found. Please also explain to the interviewer any corrections to the system specification you are making to correct those errors.

DATA FLOW DIAGRAMS

CONTEXT DIAGRAM FOR INVENTORY CONTROL SYSTEMS



DATA FLOW DIAGRAM FOR INVENTORY CONTROL SYSTEMS

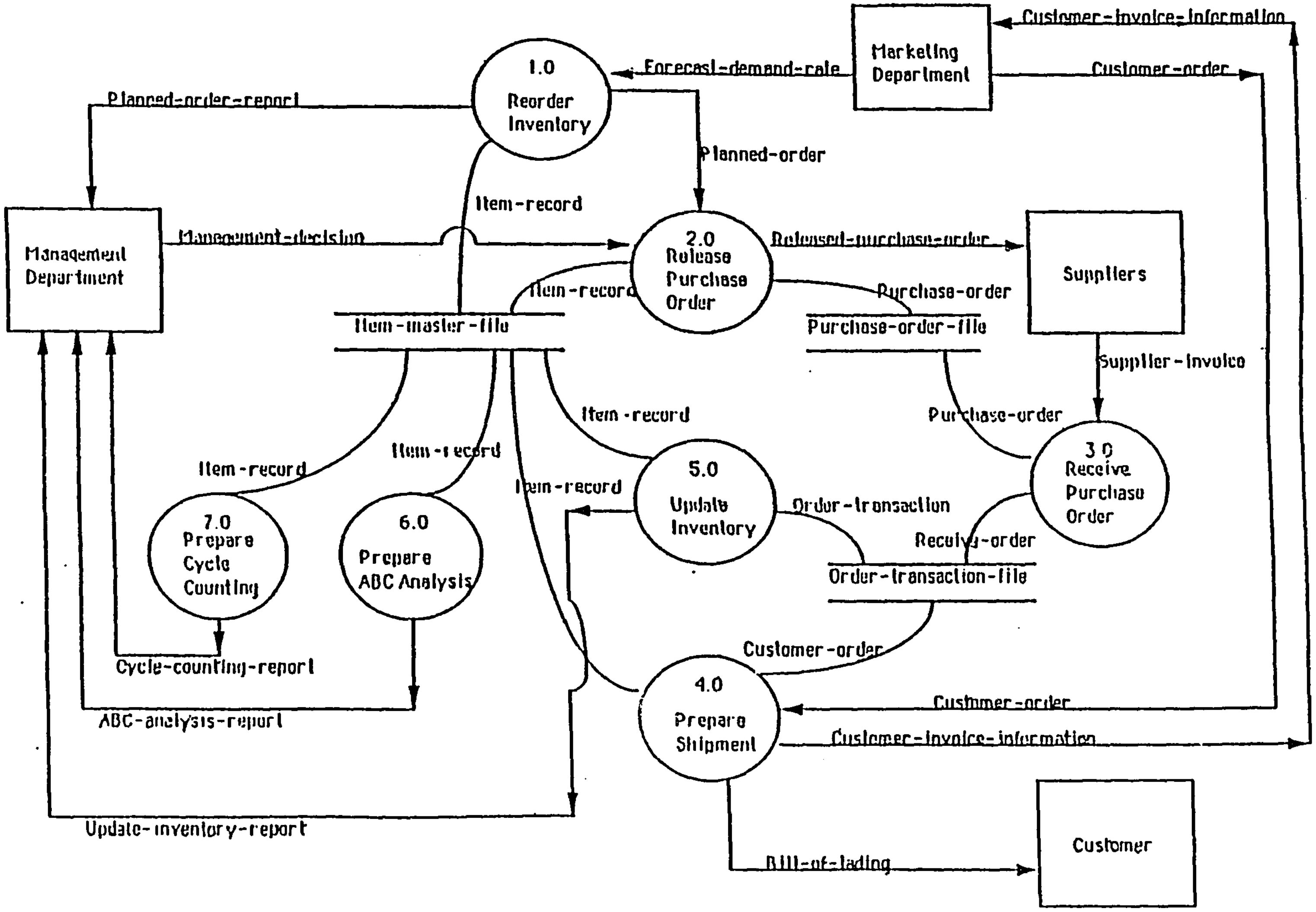


Diagram 1.0
Reorder-Inventory-Process

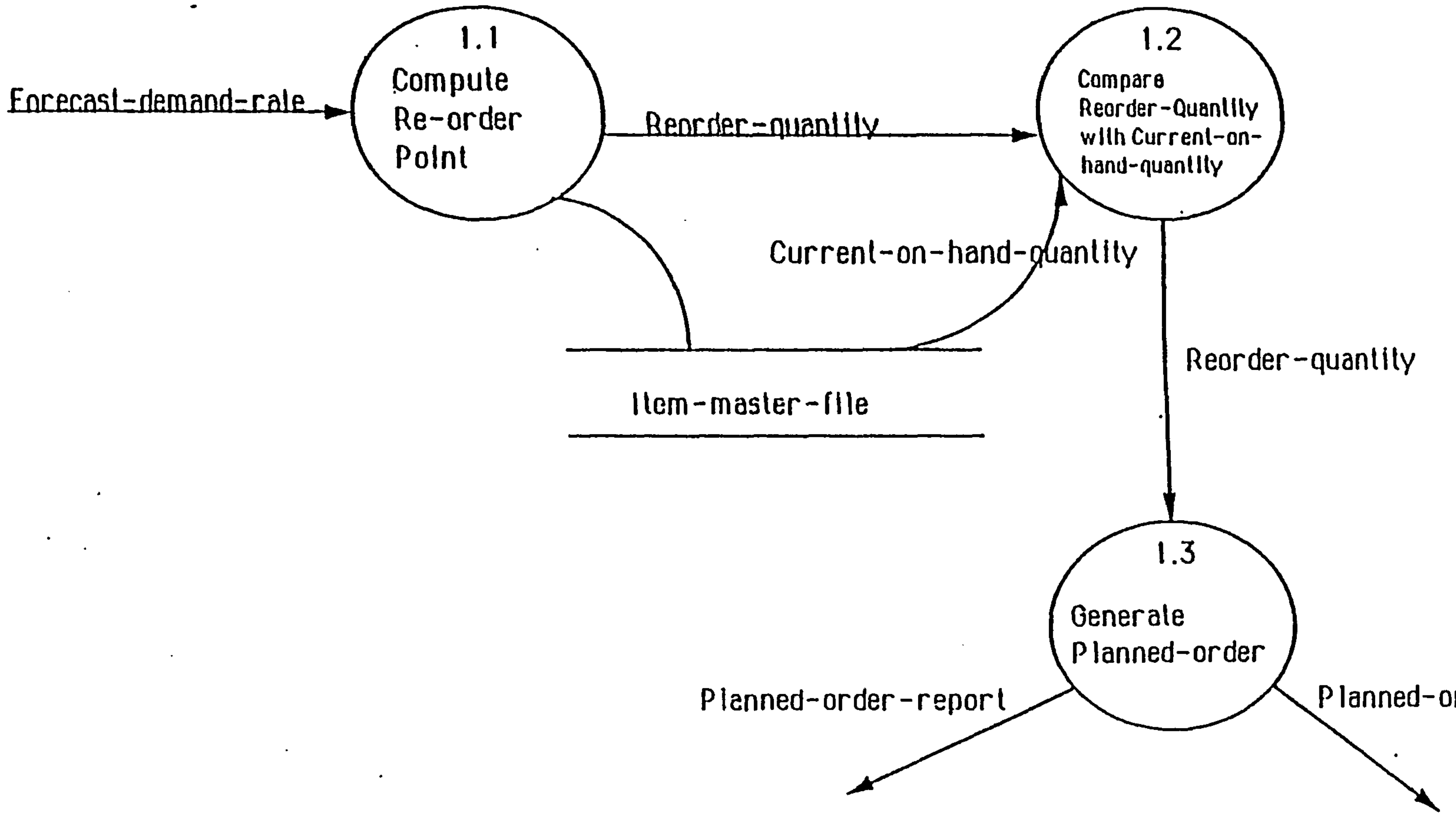


Diagram 2.0
Release-Purchase-Order-Process

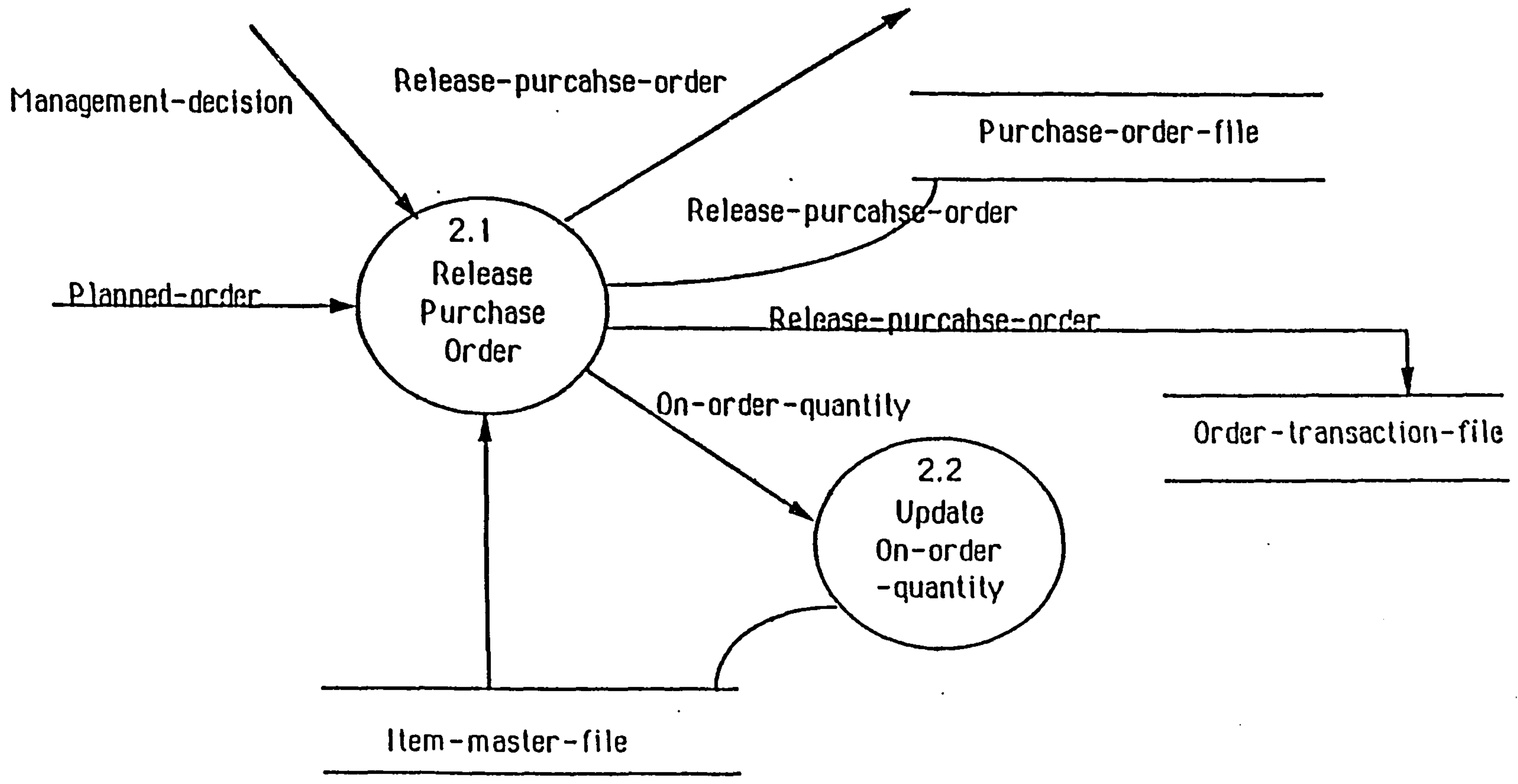


Diagram 3.0
Recieve-Purchase-Order-Process

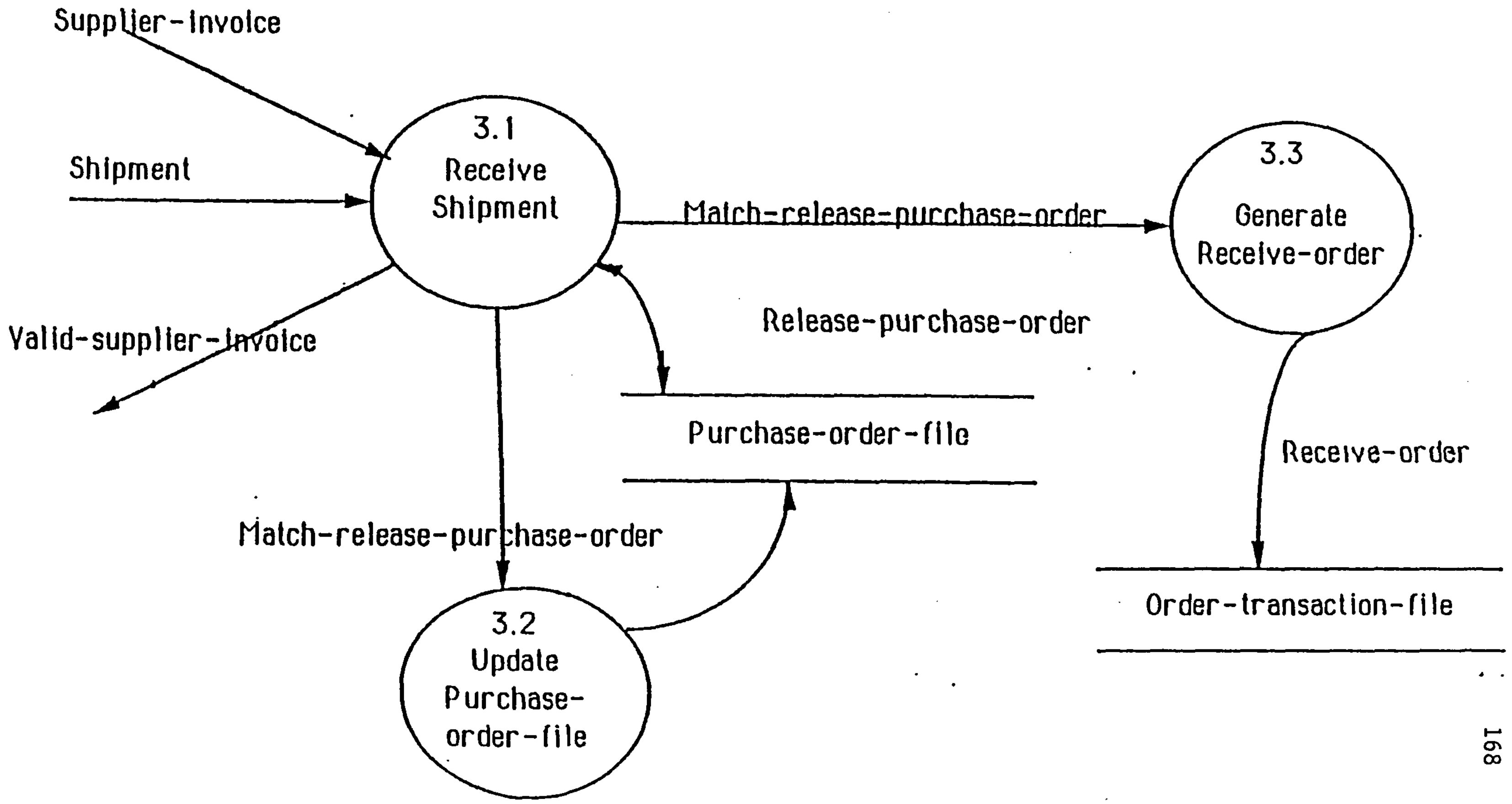


Diagram 4.0
Prepare-Shipment-Process

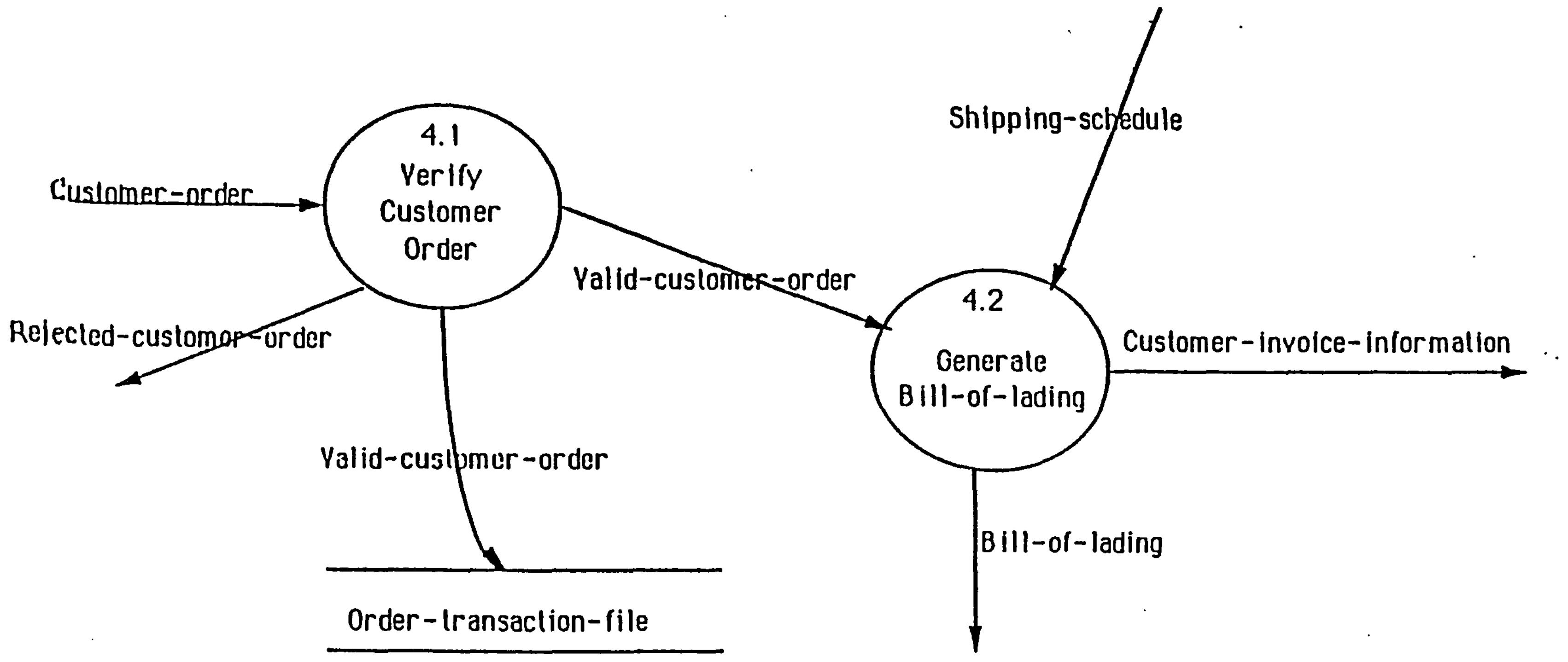


Diagram 5.0
Update-Inventory-Process

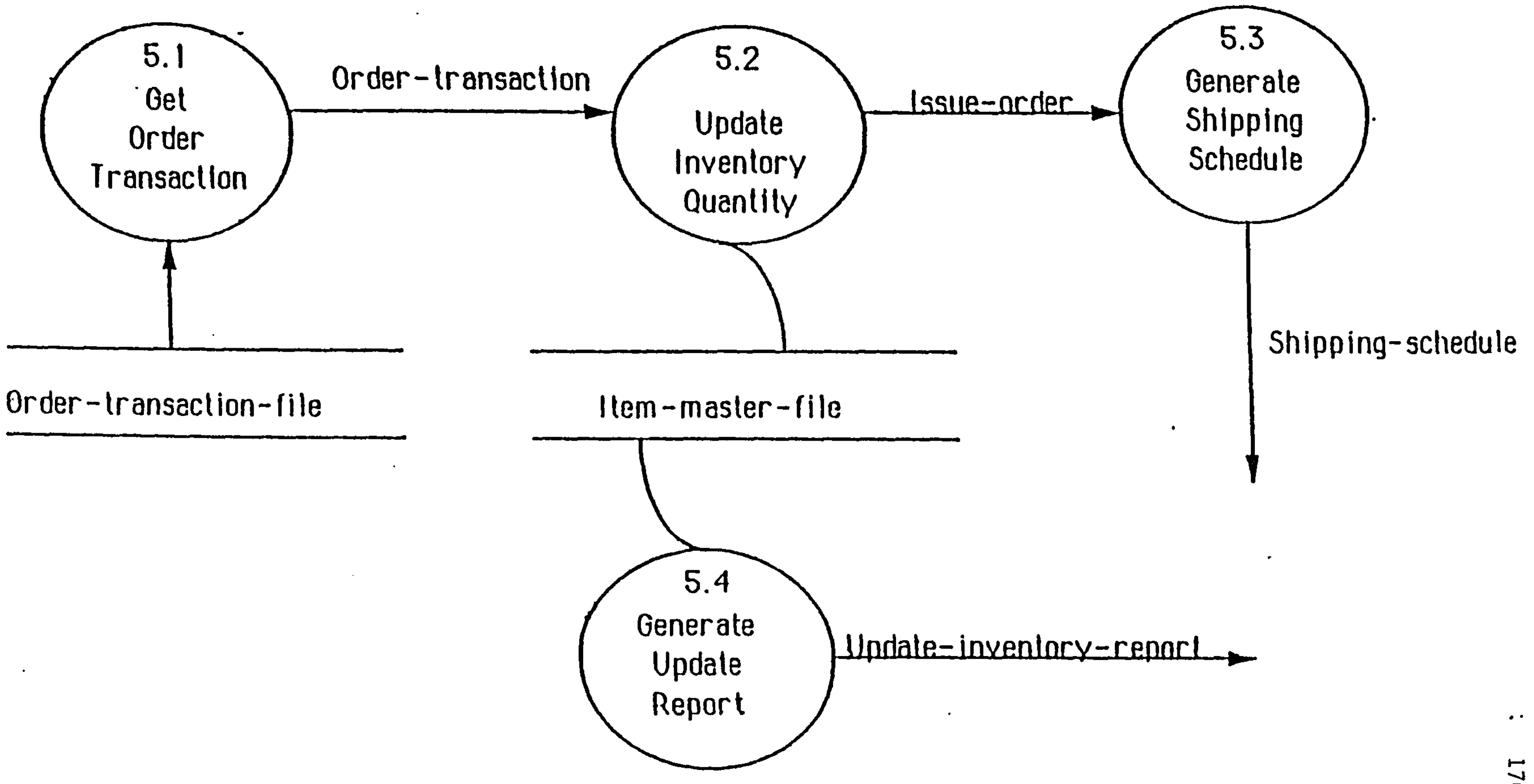


Diagram 5.2
Update-Inventory-Quantity-Process

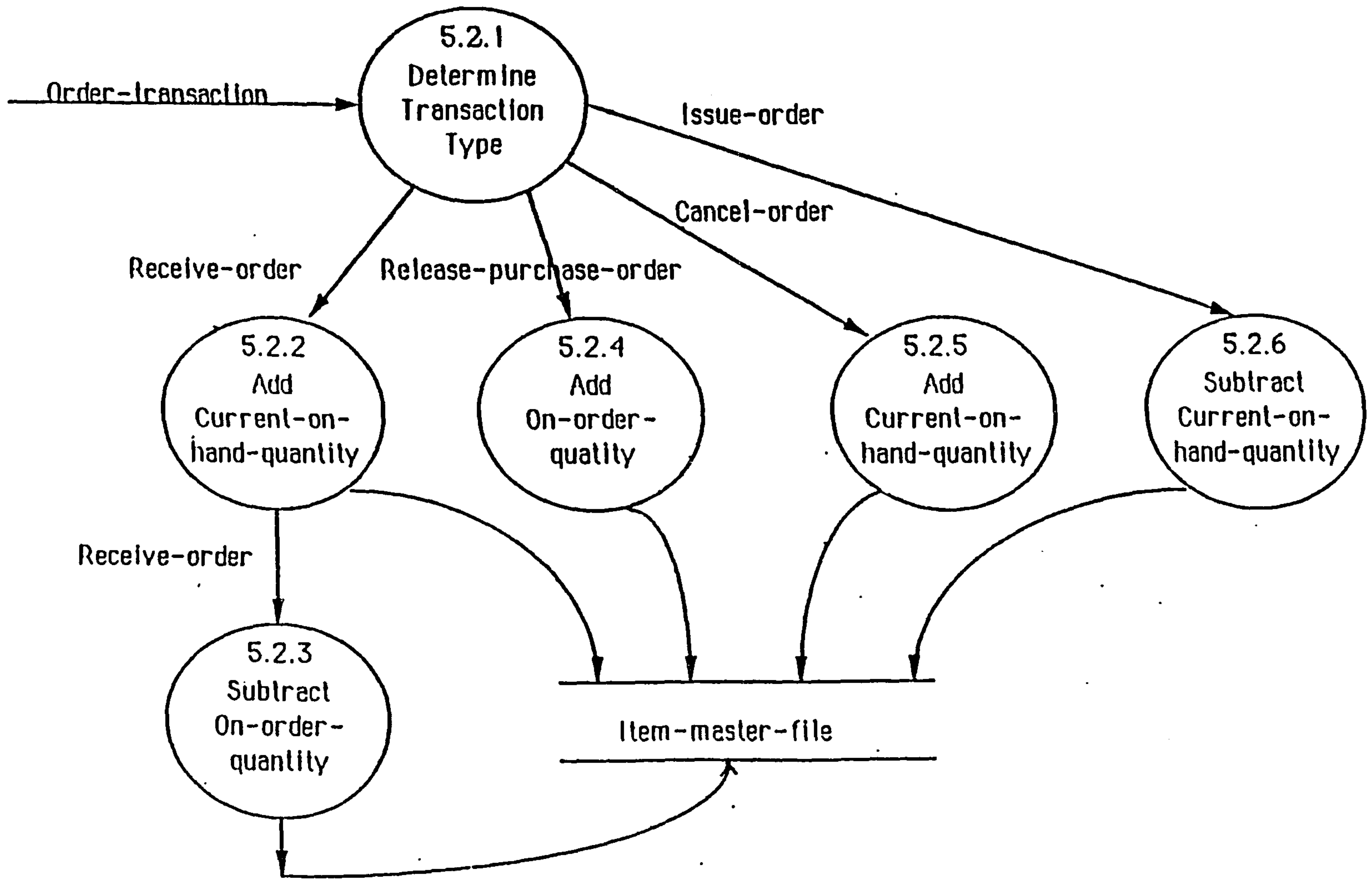


Diagram 6.0
Prepare-ABC-Analysis-Process

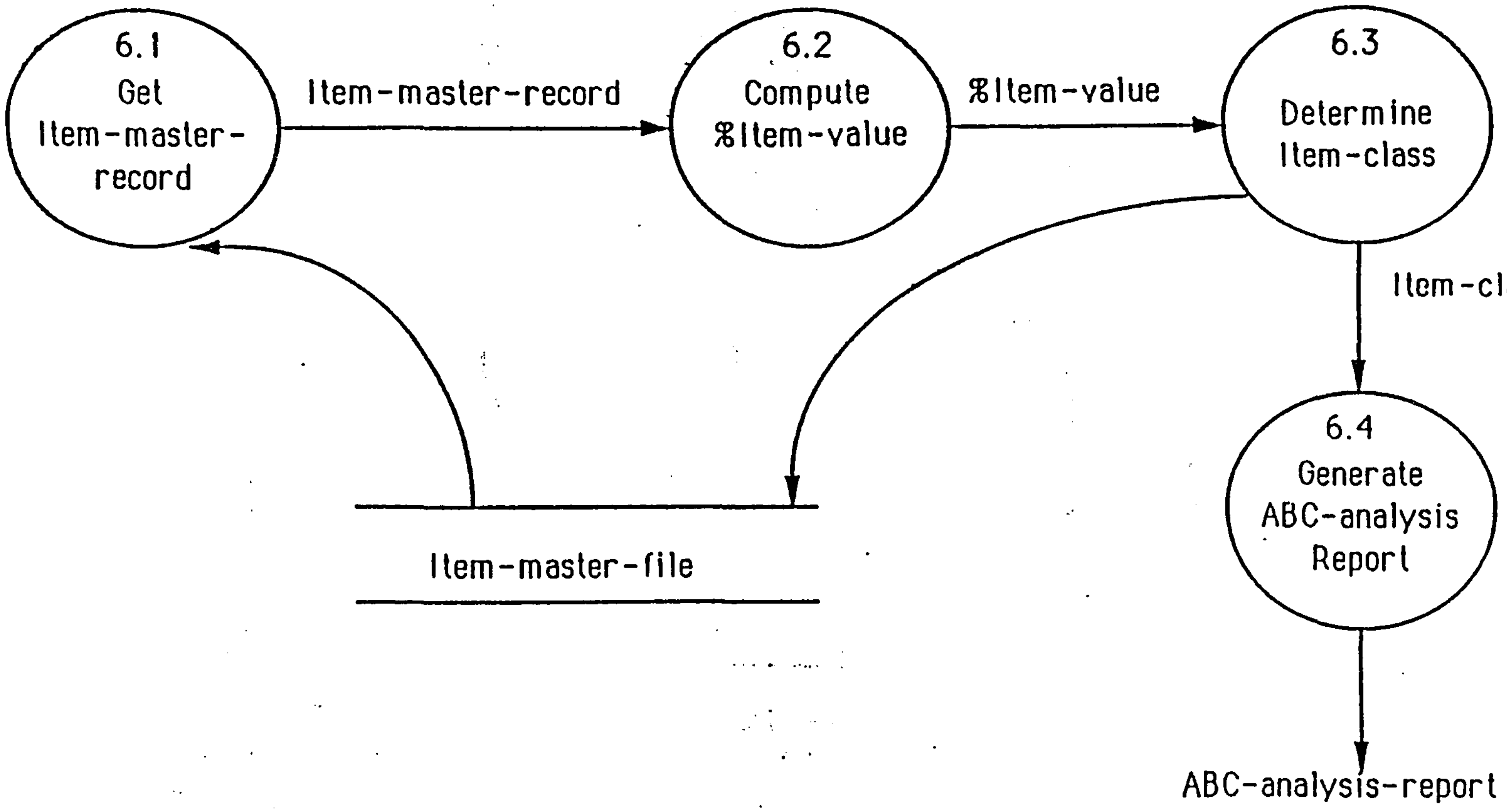
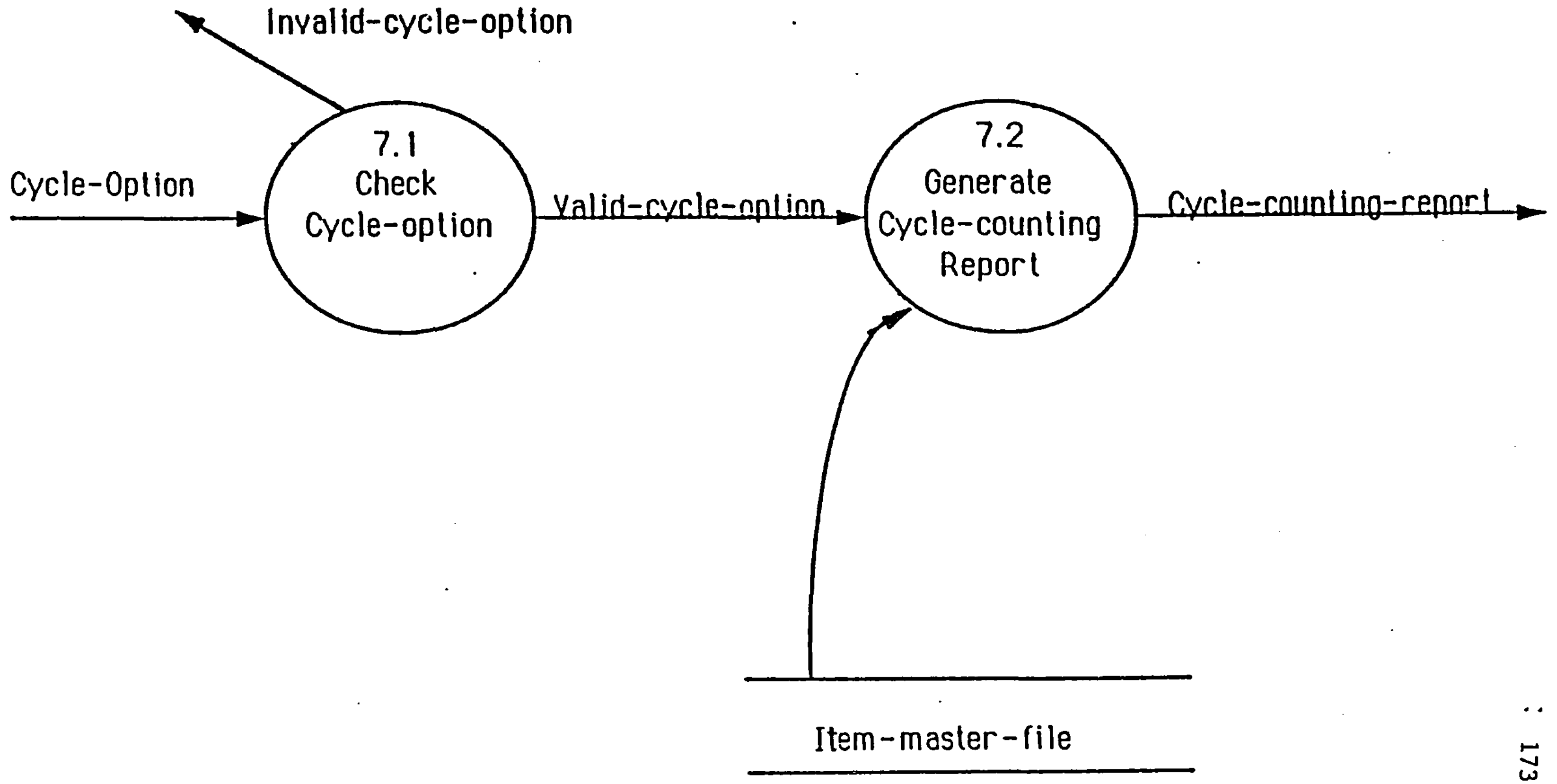


Diagram 7.0
Prepare-Cycle-Counting-Process



PROCESS DESCRIPTIONS

PROCESS DESCRIPTION

PROCESS NUMBER:	1.0
GRAPH NAME:	DFD-FOR-INVENTORY-CONTROL-SYSTEM
PROCESS LABEL:	REORDER-INVENTORY

EXPLOSION DIAGRAM:DIAGRAM 1.0

DESCRIPTION: The "Forecast-demand-rate" is received from the Marketing-department everyday in the morning. The "Reorder-quantity" is computed and compared with the "Current-on-hand-quantity" for each item in "Item-master-file". If the "Reorder-quantity" is greater than the "Current-on-hand-quantity", the "Reorder-quantity" will be used to generate the "Planned-order" and the "Planned-order-report" is sent to the management department for a review and decision to order or cancel each "Planned order".

PROCESS DESCRIPTION

PROCESS NUMBER:	1.1
GRAPH NAME:	DIAGRAM 1.0
PROCESS LABEL:	COMPUTE-RE-ORDER-POINT

DESCRIPTION: The "Reorder-quantity" for each item in the "Item-master-file" is calculated by multiplying the "Forecast-demand-rate" by the "Lead-time" plus the "Safety-stock". The "Lead-time" and the "Safety-stock" are stored in the "Item-record" in the "Item-master-file". The "Item-master-record" is retrieved by using the "Item-number" as a key to search for the match record. The "Reorder-quantity" is forwarded to process 1.2

PROCESS DESCRIPTION

PROCESS NUMBER:	1.2
GRAPH NAME:	DIAGRAM 1.0
PROCESS LABEL:	COMPARE-REORDER-QUANTITY-WITH-CURRENT-ON-HAND-QUANTITY
DESCRIPTION:	If the "Current-on-hand-quantity" for that item is less than or equal to its "Reorder-quantity", then the "Reorder-quantity" is forwarded to the "Generate-planned-order" process or process 1.3.

PROCESS DESCRIPTION

PROCESS NUMBER:	1.3
GRAPH NAME:	DIAGRAM 1.0
PROCESS LABEL:	GENERATE-PLANNED-ORDER
DESCRIPTION:	The "Reorder-quantity" is received and the "Planned-order" for that item is generated and the "Planned-order-report" is printed and forwarded to the management department for the decision whether to release or cancel that "Planned-order" for each item.

PROCESS DESCRIPTION

PROCESS NUMBER:	2.0
GRAPH NAME:	DFD-FOR-INVENTORY-CONTROL-SYSTEM
PROCESS LABEL:	RELEASE-PURCHASE-ORDER

EXPLOSION DIAGRAM:DIAGRAM 2.0

DESCRIPTION: The "Planned-order" is received from process 1.0. The "Management-decision" is received from the management department. If the decision is to release the "Planned-order", the "Release-purchase-order" is created and saved in the "Purchased-order-file", then printed and forwarded to the designate supplier.

PROCESS DESCRIPTION

PROCESS NUMBER:	2.1
GRAPH NAME:	DIAGRAM 2.0
PROCESS LABEL:	RELEASE-PURCHASE-ORDER

DESCRIPTION: The "Management-decision" and "Planned-order" are received and matched. If the decision is to release that "Planned-order", the "Release-purchase-order" is create and saved in the "Purchase-order-file". The printed "Release-purchase-order" is forwarded to the designate supplier.

PROCESS DESCRIPTION

PROCESS NUMBER:	3.0
GRAPH NAME:	DFD-INVENTORY-CONTROL-SYSTEM
PROCESS LABEL:	RECEIVE-PURCHASE-ORDER

EXPLOSION DIAGRAM:DIAGRAM 3.0

DESCRIPTION: The "Supplier-invoice" together with the shipment is received at the receiving station inside the warehouse. The "Supplier-invoice" is matched with the "Purchased-order" in the "Purchase-order-file". If match, then, "Receive-order" is generated and forwarded to the "Order-transaction-file", else the shipment is returned to the supplier.

PROCESS DESCRIPTION

PROCESS NUMBER:	3.1
GRAPH NAME:	DIAGRAM 3.0
PROCESS LABEL:	RECEIVE-SHIPMENT

DESCRIPTION: The "Supplier-invoice" together with the shipment are received at the receiving station in the warehouse. The "Supplier-invoice" is matched against the "Release-purchase-order" in the "Purchase-order-file". If match, then, the match "Release-purchase-order" is forwarded to the process 3.2 to update the "Purchase-order-file" and process 3.3 to generate the "Receive-order" and stored in the "Order-transaction-file".

PROCESS DESCRIPTION

PROCESS NUMBER:	3.2
GRAPH NAME:	DIAGRAM 3.0
PROCESS LABEL:	UPDATE-PURCHASE-ORDER-FILE
DESCRIPTION:	The matched "Release-purchase-order" is received. The "On-order-quantity" is recalculated by subtracting the "Receive-quantity" from the pervious "On-order-quantity". The updated "Release-purchase-order" is replaced in the "Purchase-order-file".

PROCESS DESCRIPTION

PROCESS NUMBER:	3.3
GRAPH NAME:	DIAGRAM 3.0
PROCESS LABEL:	GENERATE-RECEIVE-ORDER
DESCRIPTION:	The match "Release-purchase-order" is received. The "Receive-order" is created and saved in the "Order-transaction-file" for updating the "Item-master-file" in process 5.0.

PROCESS DESCRIPTION

PROCESS NUMBER:	4.0
GRAPH NAME:	DFD-FOR-INVENTORY-CONTROL-SYSTEMS
PROCESS LABEL:	PREPARE-SHIPMENT

EXPLOSION DIAGRAM:DIAGRAM 4.0

DESCRIPTION: The "Customer-order" is received from the Marketing department. The valid "Customer-order" is stored in the "Order-transaction-file". The "Shipping-schedule" is received from process 5.0 (Update-inventory-process). The "Bill-of-lading" is created and sent to the customer. The "Customer-invoice-information" is forwarded to the Marketing department.

PROCESS DESCRIPTION

PROCESS NUMBER:	4.1
GRAPH NAME:	DIAGRAM 4.0
PROCESS LABEL:	VERIFY-CUSTOMER-ORDER

DESCRIPTION: The "Customer-order" is received and checked against the "Current-on-hand-quantity" for each item ordered. If the "Current-on-hand-quantity" is zero, then the "Customer-order" is rejected, else the valid "Customer-order" is forwarded to process 4.2.

PROCESS DESCRIPTION

PROCESS NUMBER:	4.2
GRAPH NAME:	DIAGRAM 4.0
PROCESS LABEL:	GENERATE-BILL-OF-LADING

DESCRIPTION: The valid "Customer-order" is received and checked against the "Shipping-schedule" for each order. If match, the "Bill-of-lading" is created, printed and sent together with a shipment package to the customer. The "Customer-invoice-information" is forwarded to the Marketing department.

PROCESS DESCRIPTION

PROCESS NUMBER:	5.0
GRAPH NAME:	DFD-FOR-INVENTORY-CONTROL-SYSTEMS
PROCESS LABEL:	UPDATE-INVENTORY

EXPLOSION DIAGRAM:DIAGRAM 5.0

DESCRIPTION: The "Order-transaction" is retrieved from the "Order-transaction-file". The "Item-record" is retrieved from the "Item-master-file", updated, and replaced back into the "Item-master-file". The "Shipping-schedule" is generated and forwarded to process 4.0 for preparation of shipment. The "Update-inventory-report" is printed and sent to the Management department for further analysis.

PROCESS DESCRIPTION

PROCESS NUMBER:	5.1
GRAPH NAME:	DIAGRAM 5.0
PROCESS LABEL:	GET-ORDER-TRANSACTION

DESCRIPTION: The order is retrieve from the "Order-transaction-file". There are four types of order: receive-order, release-purchase-order, cancel-order, and issue-order. Each order is forwarded to process 5.2 for proper updating procedure.

PROCESS DESCRIPTION

PROCESS NUMBER:	5.2
GRAPH NAME:	DIAGRAM 5.0
PROCESS LABEL:	UPDATE-INVENTORY-QUANTITY

EXPLOSION DIAGRAM:DIAGRAM 5.2

DESCRIPTION:

PROCESS DESCRIPTION

PROCESS NUMBER:	5.2.1
GRAPH NAME:	DIAGRAM 5.2
PROCESS LABEL:	DETERMINE-TRANSACTION-TYPE

DESCRIPTION: The "Order-transaction" is received. The transaction type is determined:
If it is "Receive-order", the transaction goes to process 5.2.2 (Add Current-on-hand-quantity) and process 5.2.3 (Subtract On-order-quantity).
If it is "Release-purchase-order", the transaction goes to process 5.2.4 (Add On-order-quantity).
If it is "Cancel-order", the transaction goes to process 5.2.5 (Add Current-on-hand-quantity).
If it is "Issue-order", the transaction goes to process 5.2.6 (Subtract Current-on-hand-quantity).
When finish, the "Item-master-file" is updated.

PROCESS DESCRIPTION

PROCESS NUMBER:	5.3
GRAPH NAME:	DIAGRAM 5.0
PROCESS LABEL:	GENERATE-SHIPPING-SCHEDULE

DESCRIPTION: The "Shipping-schedule" consists of "Issue-order" transactions is printed and forward to process 4.0 for shipment preparation.

PROCESS DESCRIPTION

PROCESS NUMBER:	5.4
GRAPH NAME:	DIAGRAM 5.0
PROCESS LABEL:	GENERATE-UPDATE-REPORT

DESCRIPTION: The "Update-inventory-report" is printed and forwarded to the Management department.

PROCESS DESCRIPTION

PROCESS NUMBER:	6.0
GRAPH NAME:	DFD-FOR-INVENTORY-CONTROL-SYSTEMS
PROCESS LABEL:	PREPARE-ABC-ANALYSIS

EXPLOSION DIAGRAM: DIAGRAM 6.0

DESCRIPTION: The "Item-record" for each item is retrieved from the "Item-master-file". The ABC value is computed and updated the "Item-master-file". The "ABC-analysis-report" is printed and sent to the Management Department.

PROCESS DESCRIPTION

PROCESS NUMBER:	6.1
GRAPH NAME:	DIAGRAM 6.0
PROCESS LABEL:	GET-ITEM-MASTER-RECORD

DESCRIPTION: The "Item-record" from the "Item-master-file" is retrieved sequentially and forwarded to process 6.2 (Compute-%item-value).

PROCESS DESCRIPTION

PROCESS NUMBER:	6.2
GRAPH NAME:	DIAGRAM 6.0
PROCESS LABEL:	COMPUTE-%ITEM-VALUE

DESCRIPTION: The "Item-total-value" is calculated by multiplying the "Unit-cost" by the "Customer-order-allocation".
The "Accumulated-total-value" is computed by adding the "Item-total-value" to the previous "Accumulated-total-value".
The "Percent-Item-value" is, then, computed by dividing the "Item-total-value" by "Accumulated-total-value".
Then, the "%Item-value" is forwarded to process 6.3 (Determine-item-class).

PROCESS DESCRIPTION

PROCESS NUMBER: 6.3
GRAPH NAME: DIAGRAM 6.3
PROCESS LABEL: DETERMINE-ITEM-CLASS

DESCRIPTION: If 0.0 < Percent-Item-value <= 0.2, then "Item-class" is equal to "Inexpensive-item". If 0.2 < Percent-Item-value <= 0.8, then "Item-class" is equal to "Less-important-item". If 0.8 < Percent-Item-value, then "Item-class" is equal to "Important-item". The "Item-class" is forwarded to process 6.4 (Generate-ABC-analysis-report).

PROCESS DESCRIPTION

PROCESS NUMBER: 7.0
GRAPH NAME: DFD-FOR-INVENTORY-CONTROL-SYSTEMS
PROCESS LABEL: PREPARE-CYCLE-COUNTING

EXPLOSION DIAGRAM:DIAGRAM 7.0

DESCRIPTION: The "Item-record" is retrieved from the "Item-master-file" sequentially. The "Cycle-option" is checked and used to generate the "Cycle-counting-report" which will be sent to the Management department for further analysis.

PROCESS DESCRIPTION

PROCESS NUMBER:	7.1
GRAPH NAME:	DIAGRAM 7.0
PROCESS LABEL:	CHECK-CYCLE-OPTION
DESCRIPTION:	The "Cycle-option" is checked against the time. if the "Cycle-option" is incorrect, the invalid "Cycle-option" is printed, else it is forwarded to process 7.2 (Generate-Cycle-Counting-Report).

PROCESS DESCRIPTION

PROCESS NUMBER:	7.2
GRAPH NAME:	DIAGRAM 7.0
PROCESS LABEL:	GENERATE-CYCLE-COUNTING-REPORT
DESCRIPTION:	The "Cycle-counting-report" contains all the "Item-record" that have the valid-cycle-option. The "Cycle-counting-report" is printed and sent to the Management department.

DATA DICTIONARY

DATA DICTIONARY FOR COMPLEX SYSTEM PROBLEM

ABC-analysis-report	=	{ Item-record + Item-class-code }
Accumulated-total-value		
Cancel-order	=	Customer-order
Customer-address	=	Street + City + State + Zip-code
Customer-invoice-information	=	Customer-order + Bill-of-lading
Customer-name	=	Last-name + First-name
Customer-order	=	Customer-order-number + Customer-name + customer-address + [Item-number + Order-quantity]
Customer-order-number	=	ID
Cycle-counting-report	=	{ Item-record + Cycle-counting-code }
Bill-of-lading	=	Customer-order + Stock-location + Shipping-quantity + Shipping-personnel-name
Forecast-demand-rate	=	Item-number + Forecast-quantity
Forecast-quantity		
Invalid-cycle-option		
Issue-order	=	Customer-order
Item-master-file	=	{ Item-record }
Item-number	=	ID
Item-record	=	Item-number + Item-description + Unit-of-measure + Stock-location + Unit-cost + Unit-price + Planning-data [Current-on-hand-quantity + On-order-quantity + Customer-order-allocation] + Supplier-number + Item-class [Important-item, less-important-item, Inexpensive-item] + Cycle-counting-code [Daily, Weekly, Monthly, Bi-monthly, Quarterly, Semi-annual, Annual]
Item-total-value		
Management-decision	=	{ Item-number + [Release, Cancel]}
On-order-quantity	=	Reorder-quantity
Order-quantity		
Order-transaction	=	[Receive-order, Release-purchase-order, Cancel-order, Issue-order]
Order-transaction-file	=	{ Order-transaction }
Percent-Item-value		
Planned-order	=	Item-number + Reorder-quantity

DATA DICTIONARY FOR COMPLEX SYSTEM PROBLEM
(continued)

Planned-order-report	=	{ Planned-order }
Purchase-order-file	=	{ Release-purchase-order }
Receive-order	=	Item-number + Receive-quantity
Receive-quantity		
Release-purchase-order	=	Purchase-order-number
		+ Supplier-record
		+ { Item-number
		+ On-order-quantity }
Reorder-quantity		
Shipping-quantity		
Shipping-personnel-name		
Shipping-schedule	=	{ Issue-order }
Stock-location		
Supplier-address		
Supplier-invoice	=	Supplier-record
		+ { Item-number + Receive-quantity}
Supplier-name		
Supplier-number		
Supplier-record	=	Supplier-number + Supplier-name
		+ Supplier-address
Update-inventory-report	=	{ Item-record }
Valid-cycle-option		

APPENDIX E

THE ACTUAL SEEDED ERRORS IN THE SIMPLE SYSTEM DESIGN PROBLEM CASE

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Context diagram	1	Data flow	Customer Payment	Incorrect direction	From customer to the process
	2	Data flow	Billing code	Missing symbol	Add data flow symbol between process and unit charge file
	3	Data flow	Billing code	Missing name	Add data flow name "Billing-code"
	4	Data flow	Customer	Incomplete data flow name	Data flow name "Customer-record"
DFD level 1	1	Data flow	None	Missing data flow name b/w "Meter-reader" and Process 1	Add data flow name "Meter-reading"
	2	Data flow	Customer record	Incorrect direction	Change direction as from Process 4 to a Customer-file
	3	Data flow	New bill	Incorrect name	Change data flow name to "Customer-record"
	4	Data flow	None	Missing data flow name b/w Process 2 and Customer file	Add data flow name "Meter-number"
	5	Process	Compute New Bill	Missing process number	Add process number 3
Data Diction-ary (DD)	1	DD entry	Amount-used	Missing definition	Add definition as "number of cycle count"
	2	DD entry	Beginning-meter-reading	Missing definition	Add definition as "number of cycle count"
	3	DD entry	Billing-code	Missing definition	Add definition as "six digits number"
	4	DD entry	Date	Missing definition	Add definition as "month/day/year"

Appendix E (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Data Dictionary (DD)	5	DD entry	Date-next-meter-reading	Missing definition	Add definition as "number of cycle count"
	6	DD entry	Ending-meter-reading	Missing definition	Add definition as "number of cycle count"
	7	DD entry	Franchise-charge	Incorrect definition	Add definition as "percentage"
	8	DD entry	Last-date-to-pay	Incorrect definition	Add definition as "month/day/year"
	9	DD entry	Last-meter-reading	Incorrect definition	Add definition as "number of cycle count"
	10	DD entry	Meter-number	Missing definition	Add definition as "six digit number"
	11	DD entry	Unit-charge	Missing definition	Add definition as "dollar amount"
	12	DD entry	Unpaid-balance	Incorrect definition	Add definition as "dollar amount"
	13	DD entry	Utility-charge	Missing definition	Add definition as "percentage"

Total 22

APPENDIX F

THE ACTUAL SEEDED ERRORS IN THE COMPLEX SYSTEM DESIGN PROBLEM CASE

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Context diagram	1	Data flow	n/a	Missing data flow name b/w process and customer entity	Add data flow name "Bill-of-lading"
Diagram level 1	1	Data flow	Item-record	Missing an arrow head b/w Process 1 and Item-master-	Add an arrow head to data flow from Item-master-file to Process 1
	2	Data flow	Item-record	Missing an arrow head b/w Process 2 and Item-master-file	Add an arrow head to data flow from Item-master-file to Process 2
	3	Data flow	n/a	Missing data flow from Process 2 to Item-master-file	Add data flow name "Item-number"from Process 2 to Item-master-file
	4	Data flow	n/a	Missing data flow from Process 5 to Item-master-file	Add data flow name "Item-record"from Item-master-file to Process 5
	5	Data flow	Item-record	Missing an arrow head b/w Process 5 and Item-master-file	Add an arrow head to data flow from Item-master-file to Process 5
	6	Data flow	Item-record	Missing an arrow head b/w Process 4 and Item-master-file	Add an arrow head to data flow from Item-master-file to Process 4
	7	Data flow	Item-record	Missing an arrow head b/w Process 6 and Item-master-file	Add an arrow head to data flow from Item-master-file to Process 6

Appendix F (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
	8	Data flow	n/a	Missing data flow from Process 6 to Item-master-file	Add data flow name "Item-record"from Process 6 to Item-master-file
	9	Data flow	Item-record	Missing an arrow head b/w Process 7 and Item-master-file	Add an arrow head to data flow from Item-master-file to Process 7
	10	Data flow	n/a	Missing data flow from Management entity to Process 7	Add data flow name "Cycle-option" from Management entity to Process 7
	11	Data flow	n/a	Missing data flow from Process 5 to Process 4	Add data flow name "Shipping-schedule" from Process 5 to Process 4
	12	Data flow	Customer-order	Missing an arrow head b/w Process 4 and Order-transac tion-file	Add an arrow head to data flow from Process 4 to Order-transaction-file
	13	Data flow	Order-transact-ion	Missing an arrow head b/w Order-transac tion-file and Process 5	Add an arrow head to data flow from Order-transaction-file to Process 5
	14	Data flow	n/a	Missing data flow from Process 2 to Order-transac tion-file	Add data flow name "Released-purchase-order" from Process 2 to Order-transact ion-file
	15	Data flow	Received-order	Missing an arrow head b/w Order-transac tion-file and Process 3	Add an arrow head to data flow from Process 3 to Order-transaction-file
	16	Data flow	Purchased-order	Missing an arrow head b/w Purchased-order-file and Process 3	Add an arrow head to data flow from Purchased-order-file to Process 3

Appendix F (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
	17	Data flow	n/a	Missing data flow from Process 3 to Purchased-order -file	Add data flow name "Purchased-order-number"from Process 3 to Purchased-order-file
	18	Data flow	n/a	Missing data flow from Process 3 to Supplier entity	Add data flow name "Invalid-invoice" from Process 3 to Supplier entity
	19	Data flow	Purchased-order	Incorrect data flow name from Process 2 to Purchased-order -file	Correct data flow name to "Released-purchase-order"
	20	Data flow	Purchased-order	Missing an arrow head b/w Purchased-order-file and Process 2	Add an arrow head to data flow from Process 2 to Purchased-order-file
	21	Data flow	n/a	Missing data flow from Process 1 to Item-master-file	Add data flow name "Item-number" from Process 1 to Item-master-file
Diagram 1	1	Data flow	n/a	Missing data flow from Process 1.1 to Item-master-file	Add data flow name "Item-number" from Process 1.1 to Item-master-file
	2	Data flow	n/a	Missing data flow name b/w Process 1.1 and Item-master-file	Add data flow name "Item-record" from Item-master-file to Process 1.1
	3	Data flow	n/a	Missing an arrow head b/w Item-master-file and Process 1.1	Add an arrow head to data flow from Item-master-file to Process 1.1
	4	Data flow	Reorder-quantity	Incorrect data flow name b/w Process 1.1 and Process 1.2 Process 1.1	Correct name to "Computed-reorder-quantity"

Appendix F (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Diagram 2	1	Data flow	n/a	Missing data flow b/w Process 2.1 and Item-master-file	Add data flow name "Item-number" from Process 2.1 to Item-master-file
	2	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 2.1	Add data flow name "Item-record" from Item-master-file to Process 2.1
	3	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 2.2	Add data flow name "Item-record" from Process 2.2 to Item-master-file
	4	Data flow	Released-purchase-order	Missing an arrow head b/w Process 2.1 and Purchased-order-file	Add arrow head to data flow from Process 2.1 to Purchased-order-file
Diagram 3	1	Data flow	Shipment	Incorrect data flow, material flow is not allowed in data flow diagram	Delete data flow symbol and name "Shipment"
Diagram 4	1	Data flow	n/a	Missing data flow into Process 4.1	Add data flow name "Item-record" to Process 4.1
Diagram 5	1	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 5.1	Add data flow name "Order-transaction-record" from Item-master-file to Process 5.1
	2	Data flow	n/a	Missing data flow symbol b/w Item-master-file and Process 5.2	Add data flow name "Item-number" from Process 5.2 to Item-master-file

Appendix F (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Diagram 5	3	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 5.2	Add data flow name "Item-record" from Process 5.2 to Item-master-file
	4	Data flow	n/a	Missing an arrow head b/w Item-master-file and Process 5.2	Add an arrow head to data flow from Item-master-file to Process 5.2
	5	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 5.4	Add data flow name "Item-record" from Process 5.4 to Item-master-file
	6	Data flow	n/a	Missing an arrow head b/w Item-master-file and Process 5.4	Add an arrow head to data flow from Item-master-file to Process 5.4
Diagram 5.1	1	Data flow	n/a	Missing data flow into Process 5.2.1	Add data flow name "Item-record" to go into Process 5.2.1
	2	Data flow	n/a	Missing data flow b/w Process 5.2.2 and Item-master -file	Add data flow name "Item-record" from Item-master-file to Process 5.2.2
	3	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 5.2.4	Add data flow name "Item-record"
	4	Data flow	n/a	Missing an arrow head b/w Item-master-file and Process 5.2.5	Add data flow name "Item-record"
	5	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 5.2.6	Add data flow name "Item-record"

Appendix F (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Diagram 5.2	6	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 5.2.3	Add data flow name "Item-record"
Diagram 6	1	Data flow	n/a	Missing data flow b/w Item-master-file and Process 6.1	Add data flow name "Item-number" b/w Process 6.1 and Item-master-file
	2	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 6.2	Add data flow name "Item-record"
	3	Data flow	Item-master-record	Incorrect data flow name b/w Process 6.1 and Process 6.2	Correct data flow name to "Valid-item-record"
	4	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 6.3	Add data flow name "Item-record"
Diagram 7	1	Data flow	n/a	Missing data flow name b/w Item-master-file and Process 7.2	Add data flow name "Item-record" b/w Process 7.2 and Item-master-file
	2	Data flow	n/a	Missing data flow b/w Item-master-file and Process 7.2	Add data flow name "Item-number" from Process 7.2 to Item-master-file
Data Dictionary (DD)	1	DD entry	Accumulated-total-value	Missing definition	Add definition as "dollar amount"
	2	DD entry	Cycle-counting-code	Missing data entry and definition	Add data entry and definition as "one digit number"
	3	DD entry	Forecast-quantity	Missing definition	Add definition as "unit of measure"

Appendix F (continued)

Error Location	No	Error Symbol	Error Name	Error Description	Error Correction
Data Dictionary (DD)	4	DD entry	Invalid-cycle-option	Missing definition	Add definition as "cycle-counting-code"
	5	DD entry	Item-total-value	Missing definition	Add definition as "dollar amount"
	6	DD entry	Order-quantity	Missing definition	Add definition as "unit of measure"
	7	DD entry	Percent-item-value	Missing definition	Add definition as "percentage"
	8	DD entry	Receive-quantity	Missing definition	Add definition as "unit of measure"
	9	DD entry	Reorder-quantity	Missing definition	Add definition as "unit of measure"
	10	DD entry	Shipping-quantity	Missing definition	Add definition as "unit of measure
	11	DD entry	Shipping-personal-name	Missing definition	Add definition as "string of characters"
	12	DD entry	Stock-location	Missing definition	Add definition as "three characters"
	13	DD entry	Supplier-address	Missing definition	Add definition as "string of characters"
	14	DD entry	Supplier-name	Missing definition	Add definition as "string of characters"
	15	DD entry	Supplier-number	Missing definition	Add definition as "six digits numbers"
	16	DD entry	Valid-cycle-option	Missing definition	Add definition as "cycle-counting code"
Total	66				

APPENDIX G

THE EXPERIMENTAL DATA

Subject Number	Independent Variables			Dependent Variables	
	Tools	System Complexity	Analyst's Experience	Quality	Productivity
1	Manual	Simple	Less	40.90	5.14
2	Manual	Simple	Less	27.30	4.05
3	Manual	Simple	Less	18.20	2.53
4	Manual	Simple	Less	18.20	2.76
5	Manual	Simple	More	13.60	5.71
6	Manual	Simple	More	22.70	15.20
7	Manual	Simple	More	40.90	21.40
8	Manual	Simple	More	34.90	10.00
9	Manual	Complex	Less	33.30	10.38
10	Manual	Complex	Less	37.90	10.87
11	Manual	Complex	Less	33.30	6.51
12	Manual	Complex	Less	31.80	7.60
13	Manual	Complex	More	39.40	24.10
14	Manual	Complex	More	57.60	17.51
15	Manual	Complex	More	48.50	24.28
16	Manual	Complex	More	31.80	16.15
17	CASE	Simple	Less	9.10	0.96
18	CASE	Simple	Less	4.54	0.40
19	CASE	Simple	Less	9.10	0.86
20	CASE	Simple	Less	13.60	1.44
21	CASE	Simple	More	18.20	3.20
22	CASE	Simple	More	9.10	1.57
23	CASE	Simple	More	18.20	1.84
24	CASE	Simple	More	15.20	2.20
25	CASE	Complex	Less	6.67	1.20
26	CASE	Complex	Less	21.70	4.30
27	CASE	Complex	Less	6.67	1.49
28	CASE	Complex	Less	10.00	2.38
29	CASE	Complex	More	6.67	1.50
30	CASE	Complex	More	11.67	2.55
31	CASE	Complex	More	26.70	3.80
32	CASE	Complex	More	8.00	4.25

APPENDIX H

EXAMPLES OF TASK ANALYSIS REPORT

Report Number:	1
Subject code:	222-1
Total Time:	160 minutes
Tool Used:	CASE tool
Complexity:	High (complex)
Experience:	High (more)

List of Activities:

1. Subject starts reading problem description and instruction provided on a floppy diskette and IBM PC AT (5 minutes)
2. When finish, subject starts using CASE tools to review data flow diagrams via graphic feature on screen
(1 minute)
3. Subject starts at the first context data flow diagram on screen, then quickly browses through seven or eight lower level of data flow diagrams, and returns to context diagram level
(1 minute)
4. At the context diagram level, subject looks at external entities and try to get more information about each external entities from their description
(1 minute)
5. Subject also looks at central process on context diagram and make a comment:
(1 minute)
".....the description of the process is hard to read on the screen. It is also in a narrative form. It can be indented....."
6. On process description screen, subject attempts to rearrange process description paragraphs into an indented form which he/she feels very comfortable to read and understand
(1 minute)
7. When finish, subject saves all of changes on reports, processes and outputs descriptions
(1 minute)
8. Subject exits graphic feature
9. Subject enters CASE tools analysis feature, then comments:
(1 minute)
".....I want to get some reports and see what sort of things exist in this system specification."
10. When subject finds no report, then comments:
".....No report! I need to add one."
11. Subject enters report option on screen, creates new report format, executes report selected option, and displays report design on screen, then, exits report option
(4 minutes)
12. Subject enters an analysis preparation feature, selects data flow diagram analysis option, then generates a summary of data flow diagram analysis and displayed the results on screen, and make comment:
(2 minutes)
".....I want to perform a multi-tasking while waiting for output from analysis preparation."
13. Subject exits analysis preparation feature and reenters to analysis feature and selects report option, then attempts to execute reports and finds entity lists are not available
(2 minutes)

14. Entity list is created and saved. A report for entity list is unsuccessfully generated. Subject decides to exit report option
(6 minutes)
15. Subject exits analysis feature
16. Enter data dictionary feature
17. Subject looks at existing data dictionary list
(1 minute)
18. Data dictionary list and report is generated and printed on provided printer
(2 minutes)
19. Subject reads a hardcopy of data dictionary list and report, then comments:
(1 minute)
".....Although I can see analysis report against electronic information (on screen), I can not use screen for anything other than just get some overview. I can not see all detail information because it has several processes, data flows and data stores. I can not see all of them side by side."
20. Subject decides to exit data dictionary feature, reenters an analysis feature, and selects graph verification option.
An analysis result is generated and displayed on screen.
Subject looks at the results on screen and compares them with the print out of data dictionary report, then decides to exit a graph verification option and an analysis feature
(3 minutes)
21. Subject reenters a graphic feature and reviews a context data flow diagram, then comment:
(8 minutes)
".....On this diagram, a customer receives product and bill-of-lading. Therefore, if we look at a customer external entity, the question is should customer send a customer order to an inventory control process or to a marking department at this level."
".....I think it is a problem here. Let's assume that a customer receives product and bill-of-lading from an inventory control process and, then, let marketing department take care of customer orders."
22. Subject corrects this error right away on screen by giving a name to an unlabeled data flow as "bill-of-lading", then, provides description of the data flow and explode it into record and number of occurrences in data dictionary, then comments:
(3 minutes)
".....Let's assume that "bill-of-lading" is a correct data flow name. Bill-of-lading data flow is, then, defined as a document flow from inventory control process to customer which contains both merchandise and shipping information."
23. Subject feels comfortable with his/her new description and saves (updates) this changes on diskette
24. Subject continues on a context diagram and looks at a customer external entity and a data flow name "forecasting-demand" from a marketing department external entity.
Subject feels that he/she can not understand why "forecasting-demand" data flow is in a context data flow diagram.
Subject needs more information about this relationship and will come back to correct them later
(1 minute)
25. Subject decides to go back to an analysis feature again
26. An extended analysis option is selected.
A report on record content analysis is executed and printed on paper via provided printer
(2 minutes)
27. Subject looks at reports and compares them with data flow diagram displayed on screen

(1 minute)

28. Subject looks at a customer invoice data flow and detects that it contains order transaction record, then looks at a supplier invoice data flow from a supplier external entity to an inventory control process, and looks at a cycle-counting process
(1 minute)

29. Subject concludes that a customer invoice and supplier invoice are similar, and a cycle counting is the same as cycle-code described in the item-master file.
Since these three objects who make up exactly the same activity, they should be merged and called with the same synonym
(1 minute)

> Notice: from observer's comment.
> Subject gets confused between customer invoice and invoice received from supplier, and assumes that they are similar and decides to give them the same name. This misunderstanding may be due to subject deficiency in distinguishing their differences under a time-constraint condition.

30. Subject decides to run a full extended analysis, while waiting for the analysis results, he/she comments:
(3 minutes)

> ".....About the problem description itself, I have a question about how this company make 25% growth in their revenue when they have an incomplete analysis of their information systems. This is an example of multi-tasking (thinking) while I am waiting for the results."

31. While waiting for all analysis reports to be completely printed out on paper, subject comments:
(2 minutes)

> ".....Again, I get all reports and assume that I know what I am doing but sometime I don't. One of the tasks required for this experiment is to determine correctness, completeness and consistency. Using CASE tools, I can define completeness of specification and its consistency. But, I can not tell whether it is correct or not. I used to work on a fast computer machine. This CASE tools is very slow and not very well put together. It does not tell me what I want to know and makes me feel uncomfortable."

32. Subject reviews analysis reports on paper and detects problems, then comments:
(5 minutes)

> ".....There are problems in several records. Some of them do not have primary key. To my knowledge, every record should have a key if it is stored."

33. Subject detects more problems on reports, and comments:
(2 minutes)

> ".....Some records have primary key and its description, foreign key, labeled on one level and unlabeled on another levels. A question about this report is what does it mean. I need to do something. But the model has serious problems based on data analysis rules. Note of errors found:
>
> (1) a number of unlabeled data flows on a context diagram, this is minor problems and can be easily corrected
>
> (2) some data flows have problems with no keys and duplicated record, some data stores have index that is not stored in the record
>
> I conclude that previous data analysis done on this specification is poor and it is too large to be fixed."

34. Subject decides to take five minutes break from CASE tools, walks around the experimental room and attempts to organize ideas together
(5 minutes)

35. Subject returns and resumes the experiment

36. After several minutes of rethinking about the problems with the specification, subject comments:
(12 minutes)
".....What would I do with these problems?...I would make additional survey of current system specification on CASE tools to clarify and define these problems.
.....Start from data analysis. If data store has an index but does not appear in the record content, I must review data store "A" structure and test each elements to see whether it belongs to data store "A" or not. Then, data store "B" and "C".
.....I expect to find any data description everywhere when I am doing specification analysis. People do not work from "Top down". They don not want to stop their analysis to describe data store "A" or "B". I am paralyzed by such analysis sequence. I must work on imperfect knowledge to identify problems at their process instead of at the end product. I can not wait until complete this process because I know I will not have a complete product. If I continue this analysis, I will identify some area that may be worth for reorganization of these problems on specification and problem with business itself.
.....Next question is how do I know what changes do I need to make or correct? The size of this problem is large and full of related errors which will take time to correct all of them. I want data analysis technique that can tell me what to do, how to do backup before change and copy new ones. So, when I change it even though it is a minor problem, I can keep track of it. I need CASE tool that provide capability to do a "flip-flop" between specification, problem description, and save time on changes."
37. Subject decides to return to the extended analysis feature on CASE tool, reviews the changed specification, and comments:
(1 minute)
".....The machine is too slow."
38. Subject executes a model modification option, and comments:
(5 minutes)
".....I find no data model, no E-R diagram, then I am left with even more problems. The problem is I have model that build from business process activities not data point of view."
39. Subject feels uncomfortable and decides to stop with the following reasons and comment:
(15 minutes)
".....If I have to fix all problems and errors in this large problem, I need to do a quick trade-off analysis as earliest as possible to see whether I can fix it within a given time constrain or not, or start the analysis all over again from scratch.
.....I need to identify the scope, run all reports as many as I can, list all names, and then check it with user requirement and real problems. I need the layout of the entire scope and problem on one screen. I need tools that can fix minor errors, not just detect them, and save my time.
.....If I have to fix this problem, I will take approximately one week to look at all previous specification and analysis results, and compare tools, models, people and their skills. I will charge at least $30 per hour for 40 hours, a total of $1,200 just to tell how long I will take to fix all problems given unlimited access to real users and a full support from management."
40. Subject decides to continue detecting and correcting errors found in data flow diagrams until he/she feels comfortable and decides to stop.
(60 minutes)

APPENDIX H (Continued)

Report Number:	2
Subject code:	122-1
Total Time:	135 minutes
Tool Used:	CASE tool
Complexity:	Low (simple)
Experience:	High (more)

List of Activities:

1. Subject starts reading problem description and instruction provided on a floppy diskette and IBM PC AT (5 minutes)
2. When finish, subject starts using CASE tools to review the first context data flow diagrams via graphic feature on screen (7 minutes)
3. Subject finds a few errors and corrects them right away. Subject changes format of entity label in order to make it easier to see and understand. Subject checks each external entities, data flows, and process on context data flow diagram such as Meter- reader, Compute-new-bill, Customer-payment, Customer- monthly-statement, Unit-charge-record, and Meter-number, then decides to explode a context data flow diagram to lower level via graphic feature to get more information (20 minutes)
4. At the first level of data flow diagram, subject checks data dictionary for Compute-new-bill element, then decides to list all of data flow diagrams and their description provided on diskette. Subject looks at the list and description on screen and, again, checks data dictionary for process on context data flow diagram, then finds no errors at context diagram level (5 minutes)
5. Subject explodes context data flow diagram into lower level, looks and changes description of the billing process, then returns to context data flow diagram level (2 minutes)
6. Subject corrects customer external entity description, meter-reader data flow, customer-payment description, then explodes a "Payment-record" and describes "Customer-payment" data flow (5 minutes)
7. When finish, subject adds "Taxes and Franchise-charge", "Customer-balance" and "Grand-total" to the "Customer-payment" data flow description (5 minutes)
8. Subject detects error with Meter-reading data flow, corrects it and add its description (2 minutes)
9. Subject detects next errors with "Unit-charge", "Meter-number" and "Customer" data flows, corrects each error and explodes to lower level of data flow diagram (level 1) (10 minutes)
10. Subject detects more errors and corrects them right away, starting from data flows, processes and data stores (21 minutes)
11. When finish, subject exits level 1 data flow diagram and enters lower level data flow diagram (diagram 3), detects errors and corrects one by one, then saves and returns to higher level data flow diagram (level 1) (17 minutes)
12. Subject decides to enter data dictionary feature, generates and prints a list of data flows from data dictionary (15 minutes)

13. Using a list of data flows, subject corrects name, label and description of data flows in data dictionary, and comments:
(1 minute)
"....I would eliminate all of duplication in data dictionary if I have two more hours."
14. Subject returns to graphic feature and corrects remaining errors, and comments:
(1 minute)
"....I do not want to use analysis feature for this problem because I can not understand what is going on inside once I have changed data flow diagrams."
15. When finish with rechecking with corrected errors, subject satisfies with his/her new specification and decides to stop.
(9 minutes)

APPENDIX H (Continued)

Report Number:	3
Subject code:	212-1
Total Time:	140 minutes
Tool Used:	Manual Tool
Complexity:	High (complex)
Experience:	High (more)

List of Activities:

1. Subject starts reading problem description and instruction provided on a paper
(2 minutes)
2. When finish, subject starts looking at context data flow diagram, then takes the remaining data flow diagrams apart and spreads them on the table in the sequence of data flow diagram levels and numbers
(5 minutes)
3. Subject checks context diagram against data flow diagram level 1, and performs level balancing check to see the connection to all of the remaining data flow diagrams
(3 minutes)
4. While looking at context diagram, subject detects many errors in context diagram such as missing data flows, arrow heads, external entities are not balanced with lower level diagrams, and corrects these obvious errors right away, then check for consistency and numbering between diagrams
(7 minutes)
5. When finish, subject forwards to lower level diagram (level 1), detects several errors such as missing arrow heads, data flow name, data flows between processes, process numbers, then correct these errors one by one
(13 minutes)
6. Subject enters data flow diagram 1.0, performs level balancing for errors, detects several errors in diagram 1.0, and corrects errors one by one, then forwards to data flow diagram 2.0
(5 minutes)
7. Within diagram 2.0, subject uses data flow diagram level 1 and an extra blank paper as a note for an "off-page and interface connector", then detects and corrects errors such as missing data flows between process 2.2 to Item-master-file, missing data flow into Purchase-order-file, removes Order-transaction-file from this diagram, and comments:
(3 minutes)
".....I use an off page label or put off page connector because it is confusing at this lower level of data flow diagrams to keep track of what is coming in and going out of these diagrams."
8. Subject enters diagram 3.0, checks level balancing between diagram 3.0 and data flow diagram level 1, detects several errors such as missing data flows which are displayed at higher level, missing data flow name, inconsistent data flow name and process name at lower level, and missing data flow arrow heads, then correct these errors one by one on this diagram
(14 minutes)
9. Subject enters diagram 4.0, performs level balancing, detects errors and correct them one by one, and comments: (10 minutes)
".....I do not know why shipping shedule is not coming in to process 4.0. This is level balancing problems in diagram 4.0. I am not sure why order is comming from managment and why

customer-invoice-information is here, unless it is used by managment department."

10. Subject performs similar level balancing on diagram 5.0, 6.0, and 7.0 respectively, detects many errors and corrects them one by one until subject satisfies with the specification
(24 minutes)
11. When finish with error detection and correction, subject redraws and rewrites new specification.
(54 minutes)

APPENDIX H (Continued)

Report Number:	4
Subject code:	112-1
Total Time:	30 minutes
Tool Used:	Manual Tool
Complexity:	Low (simple)
Experience:	High (more)

List of Activities:

1. Subject starts reading problem description and instruction provided on paper, and makes note on separate paper about the problem for later used without looking at data flow diagrams, data dictionary or process description
(6 minutes)
2. When finish reading problem, subject quickly browses through data flow diagrams, data dictionary and process description, then starts detecting errors on a context data flow diagram and corrects them right away, and comments:
(2 minutes)
".....I have a question about this data flow diagram. Is it a physical or logical? I assume it is a physical data flow diagram. Therefore, the meter reading is not modified. Process 1.0 may be eliminated."
3. Within a context data flow diagram, subject detects another errors and comments:
(2 minute)
".....A "Customer-payment" data flow from customer external entity to the process has a reverse direction. A "Customer-payment" data flow is not identified in problem description. I will change the direction of this data flow from process to customer external entity."
4. Subject turns to the next page, a data flow diagram level 1, and detects several errors such as unlabelled data flow to Process 2.0, missing process number for "Compute-new-bill" process, then corrects these errors on data flow diagram, and comments:
(2 minutes)
".....There is a problem between process 3.0 and 4.0, "Compute new bill" and "Prepare statement" processes. Data flow named "New-bill" from process 3.0 to 4.0 is different from "Customer-monthly-statement" at upper level data flow diagram, and why process 4.0 get customer record? I assume that customer record is part of process 3.0 and 4.0. I will consolidate the physical flow of these two processes."
5. Subject turns to the next page, a lower level data flow diagram 3.0, and checks it against data flow diagram level 1 on previous page. Subject detects errors such as duplicated data flow names "Meter-reading" from process 1.0 to process 2.0, and from process 2.0 to process 3.0, and corrects them by giving them different names with assumption that they must be used and changed for computing the differences in the actual usage of electricity
(2 minutes)
6. Within data flow diagram 3.0, subject detects other errors and comments:
(5 minutes)
".....Process 3.4 computes an amount of tax. It needs input from source such as Tax file or external Tax file from upper level that provides a tax utility rate. I will add a Tax file as external file at upper level of data flow diagram."

7. Subject turns to data dictionary page and corrects data dictionary description to fit with his/her correction, and comments: (3 minutes)

   ".....This data dictionary is the similar to the description of each data element."

8. Subject turns to process description pages, and add process description according to his/her correction on data flow diagrams (5 minutes)

9. Subject returns to data flow diagram 3.0 and changes data flows and adds utility tax rate as part of customer record data flow, then returns to process description for process 3.4 and adds description for utility tax rate and its computation, and comments: (3 minutes)

   ".....If I have to redraw and clean up the specification documents, I need more writing pad or prefer sophisticated tools than paper and pencil. I have to do a lot of flipping between three or five data flow diagrams and process description or data dictionary. I wish to have a windowing environment."

10. When finish updating specification, subject feels comfortable with his/her new specification and decides to stop.

APPENDIX I

EXAMPLE OF POST-EXPERIMENTAL INTERVIEW

Subject Code: ______________
Case Problem#: ______________
Tool Used: ______________

1. Which tool would you rather use in this excercise? (check one)

   1( ) CASE tool
   2( ) Manual tool

2. Why?

3. If select CASE Tool:

   What advantages would CASE tool have given you?

   What disadvantages would CASE tool have given you?

4. If select manual Tool:

   What advantages would manual tool have given you?

   What disadvantages would manual tool have given you?

5. Other comments:

APPENDIX J

EXAMPLE OF ANALYSIS REPORT

```
Date: 24-Jun-90               LEVEL BALANCING                    Page  X
Time: 16:07
PROJECT NAME: Project

LEVEL NUMBER: 1
PARENT GRAPH NAME:  Context-diagram

Parent Process: 0
Child Type:  DFD Name:  DFD-1
```

Parent INPUTS not matched on child level

TYPE	ID CARRIED IN	FLOW	ID
ELEMENT	Item-number	DATA	Customer-order
ELEMENT	Customer-order-number	DATA	Customer-order
ELEMENT	Order-quantity	DATA	Customer-order
ELEMENT	Customer-last-name	DATA	Customer-order
ELEMENT	Customer-first-name	DATA	Customer-order
ELEMENT	Customer-street	DATA	Customer-order
ELEMENT	Customer-city	DATA	Customer-order
ELEMENT	Customer-state	DATA	Customer-order
ELEMENT	Customer-zip-code	DATA	Customer-order
ELEMENT	Forecast-quantity	DATA	Forecast-demand-rate
ELEMENT	Supplier-number	DATA	Supplier-invoice
ELEMENT	Supplier-number	DATA	Supplier-invoice
ELEMENT	Supplier-address	DATA	Supplier-invoice
ELEMENT	Received-quantity	DATA	Supplier-invoice

Parent OUTPUTS not matched on child level

TYPE	ID CARRIED IN	FLOW	ID
ELEMENT	Item-number	DATA	Planned-order-report
ELEMENT	Reorder-quantity	DATA	Planned-order-report
ELEMENT	Release	DATA	Management-decision
ELEMENT	Cancel	DATA	Management-decision
ELEMENT	Customer-order-number	DATA	Customer-invoice-informat
ELEMENT	Order-quantity	DATA	Customer-invoice-informat
ELEMENT	Customer-last-name	DATA	Customer-invoice-informat
ELEMENT	Customer-first-name	DATA	Customer-invoice-informat
ELEMENT	Customer-street	DATA	Customer-invoice-informat
ELEMENT	Customer-city	DATA	Customer-invoice-informat

APPENDIX J (Continued)

```
Date: 24-Jun-90                   LEVEL BALANCING                    Page  X
Time: 16:11
PROJECT NAME: Project
```

Graph Object Summary

OBJECT TYPE	I/L	OR LABEL	NOT DESCRIBED	CHILD TYPE N/A	CHILD NOT FOUND	IN BALANCE
PROCESS	I	1.0				X

LEVEL NUMBER: 3
PARENT GRAPH NAME: Diagram 1.0

Graph Object Summary

OBJECT TYPE	I/L	OR LABEL	NOT DESCRIBED	CHILD TYPE N/A	CHILD NOT FOUND	IN BALANCE
PROCESS	I	1.1		X		
PROCESS	I	1.2		X		
PROCESS	I	1.3		X		

LEVEL NUMBER: 3
PARENT GRAPH NAME: Diagram 2.0

Graph Object Summary

OBJECT TYPE	I/L	OR LABEL	NOT DESCRIBED	CHILD TYPE N/A	CHILD NOT FOUND	IN BALANCE
PROCESS	I	2.1		X		
PROCESS	I	2.2		X		

LEVEL NUMBER: 3
PARENT GRAPH NAME: Diagram 3.0

Graph Object Summary

OBJECT TYPE	I/L	OR LABEL	NOT DESCRIBED	CHILD TYPE N/A	CHILD NOT FOUND	IN BALANCE
PROCESS	I	3.1		X		
PROCESS	I	3.2		X		
PROCESS	I	3.3		X		

APPENDIX J (Continued)

```
Date: 24-Jun-90               EQUIVALENT RECORDS                  Page  X
Time: 16:27
PROJECT NAME: Project
```

DESCRIPTION: This report lists pairs of top-level records with logically equivalent contents. Records A and B have the same lowest-level contents, though they may group and order those contents differently.

RECORD A	IS EQUIVALENT TO	RECORD B
Cycle-counting-report		Item-master-file
Cycle-counting-report		Update-inventory-report
Item-master-file		Update-inventory-report

```
Date: 24-Jun-90               FOREIGN KEYS (ALL LEVELS)           Page  X
Time: 16:33
PROJECT NAME: Project
```

DESCRIPTION: This report lists each record whose key occurs in one or more other records and is therefore a foreign key. The left hand column lists each original record and its key (indented). The right hand column lists each record that contains the origin key as either key or nonkey elements.

KEY IS FOREIGN KEY IN	RECORD NAME (S)
(REC) Supplier-invoice (ELE 3) Receive-quantity	Order-transaction

BIBLIOGRAPHY

Acly, E. (1988, March). Looking beyond CASE. IEEE Software, pp. 39-45.

Adelson, B., & Soloway, E. (1985, November). The role of domain experience in software design. IEEE Transactions on Software Engineering. pp. 1351-1360.

Adelson, B. (1984). When novices surpass experts: the difficulty of a task may increase with expertise. Journal of Experimental Psychology: Learning, Memory, and Cognition. 10(3), 483-495.

Alavi, m. (1985). High productivity alternatives for software development. Journal of Information Systems Management, 2(4), 19-24.

Albrecht, A. J. (1979). Measuring application development productivity. Proceedings on IBM Applications Development Symposium, GUIDE Int. and SHARE Inc., IBM Corp. Montrerey, California, 83.

Alloway, R. M., & Quillard, J. A. (1983, June). User manager's systems needs, MIS Quarterly, pp. 27-41.

Arthur, L. J. (1985). Measuring programmer productivity and software quality. Wiley-Interscience.

Banker, D. R., Datar, S. M., & Zweig, D. (1989). Software complexity and maintainability. Proceedings of the Tenth International Conference on Information Systems, Boston, Massachusetts, 247-256.

Basili, V. R., & Perricone, B. T. (1984, January). Software errors and complexity. Communications of the ACM, pp. 42-50.

Basili, V. R., & Zelkowitz, M. (1978). Analyzing medium scale software development. Proceeding of the 3rd International Conference on Software Engineering, IEEE. 116-123.

Bastani, F. & Iyengar, S. (1987, March). The effects of data structures on the logical complexity of programs. Communications of the ACM, pp. 250.

Batra, D., & Davis, J. G. (1989). A study of conceptual data modeling in database design: Similarities and differences between expert and novice designers. Proceedings of the Tenth International Conference on Information Systems, Boston, Massachusetts, 91-100.

Benbasat, I. & Vessey, I. (1980, June). Programming and analyst time/cost estimation. MIS Quarterly, pp. 31-43.

Beruvides, M. G., & Sumanth, D. J. (1987). Knowledge work: A conceptual analysis and structure. In D. J. Sumanth (Ed.), Productivity management frontier (pp. 127-137). Amsterdam: Elsevier Science Publishers.

Betts, M. (1987, January 19). Firm readies automation tools. Computerworld, pp. 13.

Boehm, B. W. (1981). Software engineering economics. New Jersey: Prentice-Hall.

Boehm, B. W. (1987, September). Improving Software Productivity. IEEE Computer. pp. 43-57.

Boehm, B. W. (1984, January). Verifying and validating software requirements and design specification. IEEE Software, pp. 75-88.

Brooks, F. P., (1987, April). No silver bullet: essence and accidents of software engineering. Computer, pp. 10-19.

Bubenko, J. A. (1986). Information system methodologies: A research view. In T. W. Olle, H. G. Sol, & A. A. Verrijn-Stuart. (Eds). Information systems methodologies: A framework for understanding (pp. 289-313). North-Holland: Elsevier Science.

Case, A.F. (1985, Fall). Computer-Aided Software Engineering (CASE): Technology for improving software development productivity. Data Base, pp. 35-42.

Chen, P. (1989, April). The entity-relationship approach. Byte, pp. 230-233.

Chikofsky, E. J., & Rubenstein, B. L. (1988, March). CASE: Reliability engineering for information systems. IEEE Software, pp. 11-17.

Chikofsky, E. J. (1988, March). Software technology people can really use. IEEE Software, pp. 8-10.

Chrysler, E. (1978, June). Some basic determinants of computer programming productivity. Communications of the ACM, pp. 472-483.

Conner, A. J., & Case, A. F. (1986, July 9). Making a case for CASE. Computerworld, C.W. Communication.

Constantine, L. L. (1989, April). The structured-design approach. Byte, pp. 232-233.

Corkery, M. (1986, November/December). XL/Tutor: 1986 Excelerator user survey results. Intechniques, Index Technology Corporation, 4.

Curtis, B., Krasner, H. & Iscoe, N. (1988, November). A field study of the software design process for large system. Communication of the ACM, pp. 1268-1287.

Davis, G. B., & Olson, M. H. (1984). Management Information Systems: Conceptual foundations, structure, and development. New York: McGraw-Hill.

DeMarco, T. (1978). Structured Analysis and System Specification, New York: Yourdon Inc.

Doe, D., & Bersoff, E. (1986, November). The software productivity consortium (SPC): An industry initiative to improve the productivity and quality of mission-critical software. Journal of System and Software, 6, 4, 367-378.

Ehrlich, W. K., lee, K. S., & Molisani, R. H. (1990, March). Applying reliability measurement: A case study. IEEE Software. pp. 56-64.

Eilon, S., Gold, B., & Soesan, J. (1976). Applied productivity analysis for industry. New York: Pergamon Press.

Eilot, L. B. (1985). An investigation of information requirements determination and analogical problem solving, PhD. dissertation, University of Southern California.

Emory, C. W., (1976). In R. B. Fetter, & C. McMillan (Eds). Business research methods. Illinois: Richard D. Irwin.

Esterling, R. (1980, March). Software manpower costs: A model. Datamation, pp. 164-170.

Fickas, S. & Nagarajan, P. (1988, November). Critiquing software specification. IEEE Software, pp. 37-46.

Fraser, M., Kumar, K., Vaishnavi, V. (1991). Informal and formal requirements specification languages: bridging the gap (Tech. rep. no. CIS-90-004). Atlanta: Georgia State University, Department of Computer Information systems.

Fromkin, H. L., & Streufert, S. (1983). Laboratory Experimentation. In M. D. Dunnette (Ed.), Handbook of industrial and organizational psychology (pp. 415-466). New York: John Wiley & Sons.

Gane, C. (1989, April). The Gane-Sarson approach. Byte, pp. 224-226.

Gane, c. & Sarson, T. (1979). Structure systems analysis: Tools and techniques. New Jersey: Prentice-Hall.

Gilb, T. (1973). Reliable EDP application design. New York: Petrocelli Books.

Gilb, T. (1977). Software Metrics. Massachusetts: Winthrop.

Glass, R. L. (1979). Software reliability guidebook. New Jersey: Prentice-Hall.

Gordon, B. D. (1988, June). Productivity gains form Computer-Aided Software Engineering. Accounting Horizons, 2.

Gradwell, D. J. L. (1987). A review of analyst workbench products. Analyst Workbenches, England: Pergamon, 63-84.

Grant-MacKay, J. M. (1987). Expert-novice problem solving behavior: a comparative study of task and technology. Unpublished PhD. dissertation, MSIS Department, University of Texas at Austin.

Guindon, R., & Curtis, B. (1988). Control of cognitive processes during software design: what tools are needed?. Communications of the ACM, pp. 263-267.

Halstead, M. H. (1977). Elements of software science. New York: Elsevier North-Holland Incorporated.

Hartog, C., & Herbert, M. (1986, December). 1985 Opinion survey of MIS managers: key issues, MIS Quarterly, pp. 351-361.

Herbert, M. & Hartog, C. (1986, November). MIS rates the issues. Datamation, 32, 22, pp. 79-86.

Hoffnagle, G. F., & Beregi, W. E. (1985). Automating the software development process. IBM Systems Journal, 24(2).

Ives, S., Hamilton, S., & Davis, G. B. (1980, September). A framework for research in computer-based management information systems. Management Science, pp. 910-934.

Jalote, P. (1989, May). Testing the completeness of specifications. IEEE Transactions on Software

Engineering. 15(5), pp. 526-531.

Jeffery, D. R. (1987). Software engineering productivity models for management information system development. In R. J. Boland Jr., & R. A. Hirschheim (Eds.). Critical Issues in Information Systems Research (pp. 113-134). New York: John Wiley & Sons Ltd.

Jeffery, D. R. (1987). The relationship between team size, experience, and attitudes and software development productivity. IEEE. pp. 2-8.

Jenkins, M. A. (1985). Research methodologies and MIS research. In E. Mumford, R. Hirschheim, G. Fitzgerald, & T. Wood-Harper (Eds.). Research Methods in Information Systems. North-Holland: Elsevier Science Publishers B.V.

Jones, T. C. (1978). Measuring programming quality and productivity. IBM System Journal, 17(1), 39-63.

Kearrey, J. K. & Sedlmeyer, R. L. (1986, November). Software complexity measurement. Communications of the ACM, pp. 1044-1051.

Keen, P. G. W. (1981). Decision support systems: a research perspective. In G. Fisk, and R. L. Sprague Jr. (Eds). Decision support systems: issues and challenges (pp. 23-44). Oxford: Pergamon Press.

Keen, P. G. W. (1981, January). Information systems and organizational change. Communications of the ACM, pp. 24-33.

Keen, P. G. W. & Scott-Morton, M. S. (1978). Decision support systems: an organizational perspective. Massachusetts: Addison-Wesley.

Kolodner, J. L. (1983). Towards an understanding of the role of experience in the evolution from novice to expert. International Journal of Man-Machine Studies, 19, 497-518.

Konsynski, B. R. (1984). Advances in information system design. Journal of Management Information Systems, 1(3), 5-32.

Konsynski, B. R. & Kotteman, J. E. (1981). Complexity measures in system development. Proceedings of the Second International Conference on Information Systems, Cambridge, Massachusetts, 173-200.

Konsynski, B. R., Kotteman, J. E., Nunamaker, J. F., Jr., and Scott, J. W. (1984). PLEXSYS-84: An integrated development environment for information systems. Journal of Management Information Systems, 1(3), 64-104.

Langefors, B. (1973). Theoretical analysis of information systems. Sweden: Auerbach.

Langle, G. B. (1988). An investigation of the effect of specific knowledge in functional areas of business on information system analysis and design. PhD. dissertation. University of Minnesota.

Larkin, J., McDermott, J., Simon, P. D., & Simon, H. A. (1980, June). Expert and novice performance in solving physics problems. Science, pp. 1335-1342.

Lew, K. S., Dillon, T. S., & Forward, K. E. (1988, November). Software complexity and its impact on software reliability. IEEE Transactions on Software Engineering. pp. 1645-1655.

Lewis, W. P., & Sier, G. H. (1983, February). The diagnosis of plant failure: A comparison of student and professional engineers. IEEE Transactions on Engineering Management, pp. 12-17.

Lucus, H. C., & Kaplan, R. B. (1976). A structured programming experiment. Computer Journal, 19, 136-138.

Lyytinen, K. & Lehtinen, E. (1984). On information modeling through illocutionary logic. In H. kangassalo (Ed.), Report of the third scandinavian research seminar on information modeling and data base management (pp.35-

118). Univeristy of Tampere, Tampere, Finland.

Mann, R. D. (1959). A review of the relationships between personality and performance in small groups. Psychological Bulletin. 56, 244-270.

March, J.G. (Ed.). (1965). Handbook of Organizations. Chicago: Rand McNally.

Marcus, L. & Nelson, R. (1989, July). Reaping CASE harvests. Datamation, pp. 31-34.

Margolis, N. (1988, September). CASE still lagging. Computerworld, pp. 23-26.

Margolis, N. (1988, October). CASE methodologies. Computerworld, pp. 118-120.

Martin, C. F. (1988, March). Second-generation CASE tools: A challenge to vendors. IEEE Software, pp. 46-49.

Martin, C. F. (1990, March). SWAT teams will play pivotal role in '90s development. PC Week, pp. 62-63.

McCabe, J. T., & Butler, C. W. (1989, December). Design complexity measurement and testing. Communication of the ACM, pp. 1415-1425.

McCabe, J. T. (1976). A complexity measure. IEEE Transactions on Software Engineering. SE-2(4), pp. 308-320.

McCall, J., Richards, P., & Walters, G. (1977). Factors in software quality. (NTIS No. AD-A049-014, 015, 055)

McClure, C. (1989). CASE is software automation. New Jersey: Prentice Hall.

McClure, C. (1989, April). The CASE experience. Byte, pp. 235-244.

Meyer, M. H., & Curley, K. F. (1989). A methodology for classifying the complexity of expert systems: A pilot study. Proceedings of the Tenth International Conference on Information Systems, Boston, Massachusetts, 31-40.

Miller, H. W. (1989, December). Quality software: The future of information technology. Journal of Systems Management, 8-14.

Mills, H. D., & Dyson, B. P. (1990, March). Using metrics to quantify development. IEEE Software, pp. 15-16.

Mumford, E. (1981). Participative systems design: structure and method. Systems, Objectives, Solutions. 1(1), 5-19.

Necco, C. R., Tsai, W. N., and Holgeson, K. R. (1989). Current usage of CASE software. Journal of Systems Management, 39, 5, 6-11.

Neter, J., Wasserman, W., & Kutner, M. (1985). Applied linear Statistical Models. Illinois: Irwin.

Newman, P. S. (1982). Towards an integrated development environment. IBM Systems Journal, 21(1), 81-107.

Norman, R. J., & Nunamaker, J. F. (1988). An empirical study of information systems professionals' productivity perceptions of CASE technology. Proceeding of the Ninth International Conference on Information Systems. Minneapolis, 111-118.

Norman, R. J., & Nunamaker, J. F. (1989, September). CASE Productivity Perceptions of Software Engineering Professionals. Communications of the ACM, pp. 1102-1109.

Olle, W. T., Hagelstein, J., MacDonald, I. G., Rolland, C., Sol, G. H., Van Asche, J. M. F., & Verrijn-Stuart, A. A. (1986). Information systems methodologies: A framework for understanding. New York: Elsevier Science.

Orlikowski, W. J. (1989). Division among the ranks: The social implications of CASE tools for the system developers. Proceeding of the Tenth International Conference on Information Systems, Boston, Massachusetts, 199-210.

Orr, K., Gane, K, Yourdon, E., Chen, P. P., & Constantine, L. L. (1989, April). Methodology: The experts speak. Byte, pp. 221-233.

Orr, K. (1989, April). The Warnier-Orr approach. Byte, pp. 221-223.

Ovideo, E. I. (1980). Control flow, data flow and program complexity. Proceedings of IEEE COMPSAC (pp. 146-152). Chicago, Illinois.

Palloto, J. (1988, November 29). Ken Orr on CASE: acceptance grows as plaform prices dip. PC week, pp. 31-34.

Pietrasanta, A. M. (1980). Appendix D: Position paper on software productivity. In R. F. Cotellesa (Ed.), Identifying research areas in the computer industry to 1995 (pp. 148). New Jersey: Stevens Institute.

Pressman, R. S. (1987). Software engineering: A practitioner's approach. New York: McGraw-Hill.

Prieto-Diaz, R. (1987). Domain analysis for reusability. IEEE, pp. 3-29.

Puncello, P. P., Torrigiani, P., Pietri, F., Burlon, R., Cardile, B., & Conti, M. (1988, March). ASPIS: A knowledge-based CASE environment. IEEE Software, pp. 58-65.

Putnam, L. (1987). A general empirical solution to the macro software sizing and estimating problem. IEEE Transaction of Software Engineering. 4(4), pp. 345-361.

QED Information Science, Inc. (1989). Critical issues in information processing management and technology (Vo. 5). Wellesley, Massachusetts: Author.

QED Information Science, Inc. (1989). Critical issues in information processing management and technology (Vo. 2). Wellesley, Massachusetts: Author.

Rettig, C. B. (1988, August). Survey quiz CASE users and vendors. Electrical Design News, pp. 277-288.

Rosenthal, R., & Rosnow, R.L. (1969). Artifact in behavioral research. New York: Academic Press.

Rubin, H. A. (1983, July). Macro-estimation of software development parameters: The Estimacs system. Softfair Proceedings, IEEE. pp. 109-118.

Ryan, A. J. (1989, October). Survey says: bigger budget, more CASE. Computerworld. pp. 120-122.

Shneiderman, B. (1977). Measuring computer program quality and comprehension. International Journal of Man-Machine Studies, 9, 465-478.

Simon, H. A. (1981). Sciences of the Artificial. Cambridge, Massachusetts: The MIT Press.

Smith, G. F. (1989, September/October). Representational effects on the solving of an unstructured decision problem. IEEE Transactions on Systems, Man, and Cybernetics. 19(5), pp. 1081-1090.

Sternberg, R. J., & Davidson, J. E. (1982, June). The mind of the puzzler. Psychology Today, pp. 37-44.

Stogdill, R.M. (1948). Personal factors associated with leadership. Journal of Psychology, 25, 35-71.

Stone, E. (1978). Research methods in organizational behavior. California: Goodyear.

Sumanth, D. J. (1987). Productivity management frontiers-I. Amsterdam: Elsevier.

Suydam, W. (1987, January 1). CASE make strides toward automated software development. Computer Design, pp. 49-58.

Symonds, A. J. (1988, March). Creating a software-engineering knowledge base. IEEE Software, pp. 50-57.

Symons, C. R., (1988, January). Function point analysis: difficulties and improvement. IEEE Transactions on Software Engineering. pp. 2-10.

Thadhani, A. J. (1984). Factors affecting programmer productivity during application development. IBM Systems Journal, 23(1), 19-35.

Turner, J. A. (1981). A method for measuring some properties of information systems. Proceedings of the Second International Conference Proceedings on Information Systems, Cambridge, Massachusetts, 259-276.

Turner, J. A. (1987). Understanding the elements of system design. In R. J. Boland Jr., & R. A. Hirschheim (Eds.). Critical Issues in Information Systems Research (pp. 97-112). New York: John Wiley & Sons Ltd.

Vessey, I. (1985). Expertise in debugging computer programs: A process analysis. International Journal of Man-Machines Studies, 23, 459-494.

Vessey, I., & Webber, R. (1984). Research on structured programming: an empiricist's evaluation/ IEEE Transactions on Software Engineering, SE-10, 4, pp. 397-407.

Vitalari, N. P. (1985, September). Knowledge as a basis for expertise in system analysis: an empirical study. MIS Quarterly, pp. 221-241.

Vitalari, N. P., & Dickson, G. W. (1983, November). Problem solving for effective systems analysis: an experimental exploration. Communication of the ACM, pp. 948-956.

Wallace, R. D., & Fujii, R. U. (1989, May). Software verification and validation: An overview. IEEE Software, pp. 10-17.

Walston, C. & Felix, C. (1977). A method for programming measurement and estimation. IBM Systems Journal. 16(1), 54-73.

Warnier, J. D. (1974). Logical construction of programs. Van Nostrand Reinhold.

Warnier, J. D. (1981). Logical construction of systems. Van Nostran Reinhold.

Wasserman, A. J., Pircher, P.A., Shewmake, D. T., and Kersten, M. L. (1986, February). Developing interactive information systems with the user software engineering methodology. IEEE Transactions on Software Engineering. pp. 326-345.

Watkins, P. R. (1981). A measurement approach to cognitive complexity and perception of information: Implications for information systems design. Proceedings of the Second International Conference on Information Systems, Cambridge, Massachusetts, 7-20.

Weick, K. E. (1965). Laboratory experiments with organizations In J. G. march (Ed.), Handbook of organizations (pp. 194-260). Chicago: Rand Mcnally.

Weissman, L. M. (1974). A methodology for studying the psychological complexity of computer programs. PhD. dissertation. University of Toronto, Canada, Unpublished.

Welke, R. (1983). IS/DSS: DBMS Support for Information Systems Development. In C.W. Holsapple, & A.B. Whinston (Ed.), Proceedings of the NATO Advanced Study Institute. (pp. 195-250). Estoril, Portugal, June 1-14, 1981. Holland/Boston: D. Reidel.

Weyuker, E. J. (1988, September). Evaluating software complexity measures. IEEE Software, pp. 1357-1365.

Whitten, J. L., Bentley, L. D., and Ho, T. I. M. (1986). System analysis and design methods. California: Mirror/Mosby College.

Wiedenbeck, S. (1985). Novice/expert differences in programming skills. International Journal of Man-Machine Studies, 23, 383-390.

Wright, P. L. (1974). The harassed decision maker: time pressures, distractions and the use of evidence. Journal of Applied Psychology, 59, 555-561.

Yeh, R. T. (1982). Requirement analysis: A management perspective. IEEE, pp. 410-416.

Yellen, R. E. (1990, April). System analyst performance using CASE versus manual methods. IEEE, pp. 497-501.

Yourdon, E. (1989). Modern structured analysis. New Jersey: Prentice-Hall.

Yourdon, E. (1989, April). The Yourdon approach. Byte, pp. 227-229.

Yourdon, E. (Ed.). (1979). Classics in software engineering. New York: Yourdon Press.

Yourdon, E., & Constantine, L. (1979). Structured design: Fundamentals of a discipline of computer program and systems design. New Jersey: Prentice-Hall.

VITA

Metta Ongkasuwan (Hiranyavasit) was born on June 8, 1958 in Chiengrai, Thailand. She is a daughter of Mr. Koon and Mrs. Aree Ongkasuwan. She has been married to Mr. Chairat Hiranyavasit since September 8, 1984. She received a Bachelor of Science degree in Chemical Engineering from Chulalongkorn University, Bangkok, Thailand in 1980. Upon her graduation, she was trained and worked with Thai Oil Refinery Company in Sriracha, Thailand. She received a Master of Business Administration degree in Information Systems from University of Baltimore, Maryland, USA in 1983. In 1984, she joined IBM corporation in Atlanta, Georgia, as a system designer and application programmer in information systems for administration and telecommunication department. While joining IBM corporation, she started working toward her doctoral program in Management of Information Systems at Georgia State University, Atlanta, Georgia. During her doctoral study, she had worked as research assistant for MRP II and Image Processing System projects. In 1989, she started her graduate teaching assistant in the Department of Computer Information Systems, Georgia State University, Atlanta, Georgia. She has joined the department of Management Information Science, California State University, Sacramento, California as an associate professor of Management Information Systems since 1990. Her permanent address is 20/70 Soi Ladproud 87, Bangkapi, Bangkok 10240, Thailand.