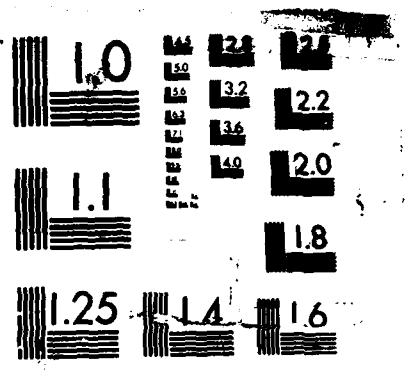
AD-A181 883 COST EFFECTIVENESS TRADE-OFFS IN SOFTWARE SUPPORT ENVIRONMENT STANDARDIZATION(U) TECHNION INTERNATIONAL INC WILMINGTON DE R WERLING 30 SEP 86 BRMC-85-5165-1 1/2

UNCLASSIFIED F33615-85-C-5165 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART

(3)

FINAL REPORT - *September 30, 1986*

TECHNION INTERNATIONAL, INC.

Cost Effectiveness TradeOffs In Software Support Environment Standardization

Contract Nr. F33615-85-C-5165

Prepared for

USAF Business Research Management Center
AFBRMC/RDCB Wright-Patterson AFB, OH

COST EFFECTIVENESS TRADE-OFFS IN ADA-BASED SOFTWARE SUPPORT ENVIRONMENT STANDARDIZATION

Richard Werling
Raymond C. Houghton, Jr.
Technion International, Inc.
P. O. Box 417
Wilmington, DE, 19899

September 30, 1986

FINAL REPORT: Contract No. F33615-85-C-5165

Prepared For:

U.S. AIR FORCE BUSINESS RESEARCH MANAGEMENT CENTER
AFBRMC/RDCB
Wright-Patterson AFB, OH
45433

HQ AIR FORCE SYSTEMS COMMAND
Deputy Chief of Staff, Acquisition Logistics
Directorate of Computer Resources,
AFSC/PLR
Andrews AFB, DC 20334

ACKNOWLEDGEMENTS

The authors are particularly indebted to several people for their help during this study. Col. Kenneth E. Nidiffer was a constant source of knowledge and insight regarding the formulation of design requirements, system specification, and economics as practiced within the Air Force and DoD. Capt. Edward C. Mitchell of AFBRMC, which made the work possible through its support and funding, was extremely helpful as we navigated through the turbulence.

William L. Byerly, of SASC Technologies, Inc. wrote the detailed software specification and programmed key aspects of the model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF ACRONYMS AND TECHNICAL TERMS

Ada	A standard programming language used in development of new major DOD systems(ANSI/MIL-STD-1815A). Ada is a registered trademark of the U.S. Government, Ada Joint Program Office.
APSE	Ada Programming Support Environment.
Benefits	In this report and in the econometric model, benefits to USAF stem from several sources: (a) reduced number and variety of environments that USAF must support; (b) use of integrated software development/support environment, incorporating advanced software tools; (c) avoidance of programming, by increased reuse of existing tested code; (d) improved computer response time, which avoids interfering with analysts and programmers work patterns; and (e) skills stemming from experience with using an integrated software environment.
Contractor	Developer of weapon systems, including the software required for their operation. Often, contractors also enhance and maintain systems and software after the initial development, and through the systems' in-service (post-deployment) life.
Costs	In this report and in the econometric model, USAF costs are differentiated into relatively "fixed" and relatively "variable" costs. Costs are further separated into three phases of an environment's life cycle: (a) the "up-front" R&D investment to design and develop an environment; b. operational costs incurred by government and contractor users throughout the country, as well as expenses for centralized support and continued development of the environment (e.g., configuration management, documentation support, quality assurance and testing); and (c) costs for continuing improvement (upgraded software versions and for eventually replacing the environment. Expenses covered include: programming and other personnel, quality assurance, training, utilities, as well as necessary expenses of supervision and management.

Econometric Model	A set of equations that describe the inter-relationships of economic factors. As used in this report, the equations (and the accompanying Lotus 1-2-3 application program for implementing them) that describe costs and benefits to USAF from use of various combinations of (a) software environments and (b) control strategies.
Environment	Advanced tools for software designers and programmers. It comprises a framework for integrating sets of methods, procedures, and computer programs (computerized software tools), to support the software life cycle.
GFE/Environment	As used in this report, a standard software development and support environment made available to contractors as government-furnished equipment.
HAPSE	Hypothetical Ada Programming Support Environment, postulated for this (and the preceding) project. The HAPSE is conceptualized as an "ideal type," to be used for investigating productivity across the software life cycle, and for comparison with actual environments as they are developed. The first level contains a basic set of software tools. The second set adds to the basic set those tools judged most necessary by software development managers. The third level contains those, plus additional tools judged valuable.
MCCR	Mission-Critical Computer Resources.
Methodology	A general collection of rules, methods, and philosophies supporting software life cycle activities.
Method	A set of specific rules, guidelines, and techniques supporting software life cycle activities.
Post-Deployment Support	Support of software after its initial deployment. During the total life cycle of a system containing software, most so-called "maintenance" is done to enhance performance of the system in which the software is embedded, by meeting new requirements or adapting to changes in other system components.
Product Differentiation	The state of the market sought by vendors, in which their products are enough different from those of other vendors that they are able to claim higher prices than would be possible if other vendors could offer competitive bids.

Productivity	The average number of software delivered source instructions (DSI) per staff work-month. Includes both freshly written and reusable software code components. Though not a perfect measure, this definition is still the <u>least unsatisfactory</u> and most widely used indicator now available.
Reliability	The probability that software will not cause failure of a system for a specified time under specified conditions.
Reusable Code	Standard proven fragments of software that can be adapted for reuse in new software systems. Reusable code provides improved reliability and maintainability as well as increased productivity, because it need not be completely redesigned, rewritten and tested.
Software	Computer programs. Also called "code."
System Life Cycle	The period of time from perception of need for a defense system that contains mission critical computer resources until its retirement. The <u>system</u> life cycle includes hardware, software, facilities, communications, policy and human elements. For contrast, see Software Life Cycle.
Software Life Cycle	As used *in this report*, the period of time from perception of need for the <u>software components</u> of a defense system. Includes initial development and post-deployment support of MCCR software. For contrast, see System Life Cycle.

EXECUTIVE SUMMARY

BACKGROUND

Purpose of Research

The objective of the research was to prepare an econometric model to determine quantitative benefits to the Air Force of implementing various strategies for controlling integrated automated software support environments. The model had to help the Air Force quantify costs and benefits obtainable by various methods of increasing productivity of developing and supporting mission-critical software.

Value to USAF. If aircraft and missiles are the heart of USAF, MCCR software is the brain. It is expensive. Every year from 1985 to 1995, the USAF will spend from $5 to $7 billion (1986 dollars) for MCCR software [Figure S-1].

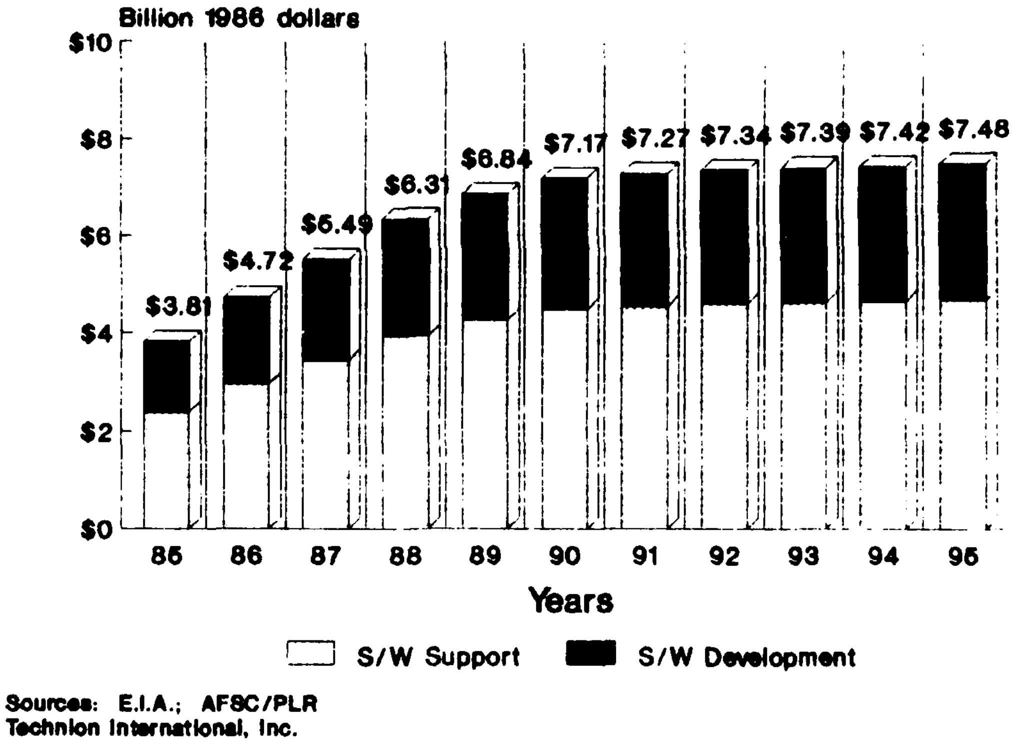


Figure S-1. USAF MCCR Software Costs.

The benefit expected as USAF uses integrated automated software support environments is large and real. Benefits to USAF will stem from: (a) reduction in the number of unique environments supported by the Air Force; (b) increases in the productivity of staff who actually use the standard environment; and (c) increased reliability and maintainability of the software produced by the standard environment.

Foundation in earlier work. The research built on work done from October 1984 to June 1985, under USAF Business Research Management Center contract Nr. F 33615-84-C-5114. The earlier research began by asking this question:

> Can and should the United States Air Force (USAF) build and supply to contractors as Government-Furnished Equipment (GFE) a Standard Government-Owned Environment (GFE/Environment) for their use in developing Ada[1]-based Computer Software for Mission Critical Systems?

The 1984-85 study produced strong qualititative indications that USAF can benefit significantly by changing ways by which its MCCR software is acquired and supported. The present contract, Nr. F 33615-85-C-5165, was undertaken to develop a mechanism for improving the quantitative definition of those benefits.

What we were asked to do

Building on the 1984-85 study, we were asked to

> " . . . prepare and validate an econometric model for implementing an Ada-based software support environment in contractor and DoD organizations. The COCOMO model will be augmented by non-quantitative sub-models.
>
> " . . . The model must include factors on control strategies and constraints on successful implementation by different organizations. The model will contain features that allow for: a) projecting productivity gains from use of various software tools and modern programming practices; b) estimating the effects of alternative control strategies for use of the environment; c) estimating effects of various constraints on successful implementation; and d) forecasting effects of providing a standard environment to contractors by exercising the model for differences in contracting strategy and contractor motivations."

We were to prepare a prototype computer program (for use on the USAF standard Zenith 100 microcomputer, and written in Microsoft FORTRAN).

[1] Ada is a registered trademark of the U.S. Government, Ada Joint Program Office (AJPO).

METHODOLOGY

Technion conducted the work as seven separate tasks. In Task 1 we developed a detailed Management Plan for the research. Subsequent task descriptions follow.

Task 2. Prepare and validate an Econometric Model for implementing an Ada-based software development/ support environment in contractor and DOD organizations. We updated data by surveying vendors, users, and developers of environments that appeared promising. While Technion began by dealing with the industry point of view as well as the Government's, the model delivered considers only the Government side. Benefits were estimated using the approach developed by Boehm in the COCOMO cost estimated model. Costs were estimated using data from the industry survey and from surveys made at Tinker AFB and by USAF Logistics Command.

Task 3. Automate the econometric model on the USAF standard Zenith Z-100 microcomputer. The resulting prototype automated model is implemented as an application of the Lotus 1-2-3 "spreadsheet" program, and is intended for use by commanders responsible for development and support of MCCR software.

Task 4. Develop the technical user Documentation on the contents of the automated model, and on how to use the automated model.

Task 5. Using the econometric model, investigate effects of providing a standard environment to contractors, based on contractors' strategy and motivations. Task 5 considered business and economic areas that are not addressed in the minimum model actually programmed. In this task we looked briefly at vendors' business concerns, such as market shares and vested interests in their own software tools, technology, and products. The major obstacle to overcome is the delay, a minimum of four years, required to develop and field a GFE Environment.

Task 6. Using the econometric model developed in Task 2, investigate effects of alternative control strategies, based on organizational constraints to successful implementation. Nine distinct control strategies are considered. They represent different combinations of voluntary [e.g., Jovial] and required [e.g., Navy's CMS-2] use of GFE environments by contractors, in both initial development and post deployment support contracts.

Task 7. Using the econometric model, select strategies and prepare cost/benefit tables for implementing environments under varied conditions. In carrying out this task, Technion first conducted sensitivity analyses of the work done in Tasks 5 and 6, then chose feasible USAF strategies for implementing environments under varied conditions.

FINDINGS

General

Results of the research make it clear that:

1. The annual USAF costs to maintain the existing massive inventory of non-standard MCCR computer programs are immense. Costs are projected by EIA to be <u>$6 billion for FY 1990</u> [EIA86], five percent of the USAF budget.

2. Few systematic measures to improve productivity for acquisition and support of MCCR software are currently in place. Technion International estimates (using as base the Electronic Industries Association's 1986 projection) that for the "business as usual" option- <u>without significant USAF action</u> - total costs to USAF will be close to <u>$6 billion)</u>, 6.7 percent of USAF budget.

3. Perhaps a dozen U.S. vendors can build an integrated, automated environment for Ada MCCR software with today's technology. The effort will require a minimum of four years and about $75 million. Such environments could harness known software engineering technology to produce <u>annual</u> improvements in productivity of <u>more than ten percent</u>. Several contractors have observed similar annual increases in productivity in their internal operations [Boeh81], [Boeh84], [Werl86]. Boehm describes systematic software productivity improvement programs at length. [Boeh81, pp. 641-689, particularly 682-689].

4. <u>By itself</u>, the GFE environment approach takes too much time. A minimum of four years (and $74 million) is needed to field a "first level" environment, capable of helping USAF improve MCCR software productivity by about six percent annually after delivery. The "third level" environment, which would help improve productivity by about ten percent annually after delivery, would need an <u>additional</u> three years (and $200 million).

5. USAF can begin improving productivity in development of MCCR software without waiting for improved software tools. Improved productivity is aided by, but is <u>not dependent on</u>, availability of an improved software development environment [Werl86]. Systematic productivity improvement efforts are well understood and clearly successful. They rely on better use of existing tools and on reuse of existing software "code fragments." [Boeh81, pp. 686].

   This approach is directed at improving the practice of MCCR software development and support, supplementing the many present efforts to further develop software engineering theory. Both are needed, but at this moment we believe USAF needs improvement in applied practice.

6. A two-pronged approach is most appropriate, with USAF undertaking <u>both</u> (a) systematic productivity improvement programs and (b) developing Ada-based software support environments. USAF may learn from recent experience with such related programs as the Army's ALS compiler.

<u>Econometric Model</u>

1. In controlling use of the integrated automated environment, two strategies offer the highest net benefits:

   (a) <u>Mandatory use for post-deployment support</u>; with <u>no</u> extraordinary requirements during development.

   (b) <u>Mandatory use for post-deployment support</u>; with <u>voluntary</u> use in development.

   For all other options, benefits are negative.

   Both of these options approximate the control strategy USAF has used for the JOVIAL language.

2. The rate of introduction of standard environments has great leverage over the magnitude of net benefits. Rapid introduction, so that use is widespread within two years, has greatest net benefit. Slow introduction, extending over four or five years, has least net benefit for USAF.

3. Net benefits to USAF are greater when existing productivity is low, less significant when productivity is high.

Sources of Benefits to USAF

Based on our research, USAF benefits would stem from these sources:

1. Increases in productivity of staff who actually develop and support MCCR software. This augmentation of human talent by software tools is the dominant economic driver. [Boeh81], [Boeh84], [Werl86].

2. Reduction in the number of unique environments now supported by USAF. Each weapon system contractor typically supplies a unique environment tailored to the weapon system supported and to the contractor's hardware, procedures, and proprietary software. Currently, USAF is supporting about 400 different languages and dialects, and several dozen unique environments [Ichb84], [Lieb86]. Other benefits include reduction of training costs and costs of contractor lock-ins. In economic terms, these are all features of "monopolistic" market relationships.

3. Increased reliability and maintainability of the software produced by the standard environment.

RECOMMENDATIONS FOR IMPLEMENTATION

External

1. Consult with appropriate Congressional committees for acceptable ways to implement these findings. An approach like that which led to the "Warner Amendment" would probably be most successful. House Government Operations Committee members have traditionally opposed any measure that seems to interfere with "full and free competition," and have challenged DoD initiatives for standardization.

In contrast, the Armed Services committees would be most likely to support USAF and DoD action to standardize software support environments. With this external support, internal USAF actions can be successful. Without it, we can expect to have new layers added to the existing "scar tissue" at DoD.

Internal

1. Competitively acquire three automated integrated software environments to develop and support MCCR software. Perhaps a dozen U.S. vendors can do this today (e.g., General Electric, TRW, Softech, Microsoft). Such environments could harness known software engineering technology to produce annual improvements in productivity of more than ten percent. (See Appendix D).

By itself, this approach takes too much time. It could be shortened if vendors reuse existing code. Starting from scratch, it would take a minimum of four years (and $50 million) to field a "first level" environment, capable of helping USAF improve MCCR software productivity by about six percent annually after delivery. The "third level" environment, which would help improve productivity by about ten percent annually after delivery, might take an additional three years (and $150 million).

2. Competitively acquire three libraries of Ada language MCCR software "fragments," for systematic reuse. This requires two efforts by each vendor: (1) test and validate candidate existing code fragments; and (2) classify and catalog them in ways that permit users to quickly identify promising candidates for reuse.

CHAPTER ONE

OBJECTIVE AND METHODOLOGY

OBJECTIVE

The objective of the research was to prepare an econometric model to determine quantitative benefits to the Air Force of implementing various strategies for controlling integrated automated software support environments. The model had to help the Air Force quantify costs and benefits obtainable by various methods of increasing productivity of developing and supporting mission-critical software.

Econometric Model

The principle deliverable is an econometric model describing costs and benefits involved in using Ada-based software development/support environments in DoD and contractor organizations. It is designed to permit comparisons among a selection of such environments. The model consists of econometric equations that describe actions required, and effects expected, during implementation. The equations were developed to describe decision-making as it typically takes place within AFSC.

Benefits. Projected benefits to USAF come from: (a) **more productivity of staff** who actually use the standard environment(s); (b) **reduction in the number of unique environments** USAF supports; and (c) increased **reliability and maintainability of software produced** using standard environment(s). In certain scenarios, annual benefits to USAF total more than $1 billion by 1995.

Costs. Projected costs to USAF accrue from: (a) **design and development** of new environment(s); (b) costs for **training government and contractor personnel** in their use; and (c) ongoing **operations and maintenance** costs. Annual costs will be less than $100 million by 1995.

Prototype computer program. We automated the model **as it was in May-June 1986**. The prototype program was written for the USAF standard Zenith Z-100 microcomputer, as an application of the LOTUS 1-2-3 computer program.

Continuing development. However, the model did not stop evolving at that point! In fact, it is still being augmented at this writing (a research phenomenon with ample precedent).

[illegible]

METHODOLOGY

Types of Econometric Model

We developed a multi-equation simulation model describing USAF support activities for MCCR software during the years 1986-1995. Before selecting this model type, we considered three frequently used general classes of econometric models. The three descriptions of economic models below are from [Pind81, pp. xv-xvi].

> "Time-Series models. In this class of models we presume to know nothing about the real world causal relationships that affect the variable we are trying to forecast. Instead we examine the past behavior of a time series in order to infer something about its future behavior. The time-series method used to produce a forecast might involve the use of a simple deterministic model such as a linear extrapolation or the use of a complex stochastic model for adaptive forecasting. . . . Models such as this have been developed and used to forecast the demand for airline capacity, seasonal telephone demand, [and] the movement of short-term interest rates. . ."

Figure I-1 shows such a forecast, which applies to 30 MCCR-like projects listed in the "COCOMO" project data base [Boeh81, pp. 498-499]. During the decade of the 1970's, productivity for development of software increased at more than 20 percent each year.

> "Single-equation regression models. In this class of models the variable under study is explained by a single function (linear or nonlinear) of explanatory variables. The equation will often be time-dependent (i.e., the time index will appear explicitly in the model), so that one can predict the response over time of the variable under study to changes in one or more of the explanatory variables. . . . often used to forecast not only the movement in short- and long-term interest rates but also many other economic and business variables."

Figure I-1 also illustrates this type of model, with "Time" being the single independent variable.

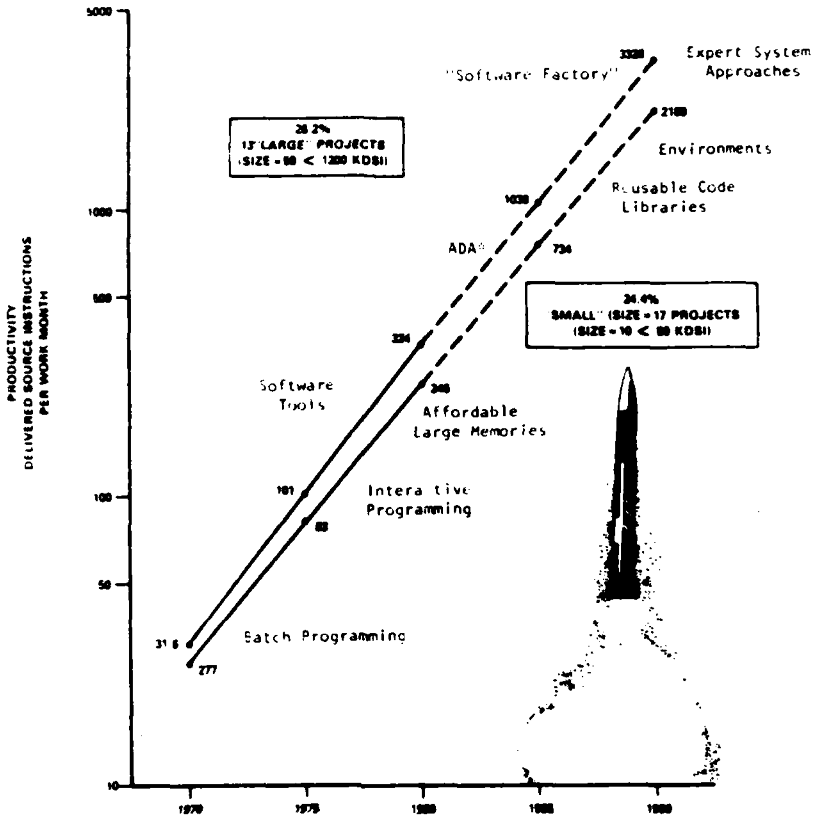


Figure I-1. Software Productivity Increases Exponentially.

"Multi-equation simulation models. In this class of models the variable to be studied may be a function of several explanatory variables, which now are related to each other as well as to the variable under study through a set of equations. The construction of a simulation model begins with the specification of a set of individual relationships, each of which is fitted to available data. Simulation is the process of solving these equations simultaneously over some range in time.

"An example of a multi-equation simulation model would be a complete model of the United States textile industry that contains equations explaining variables such as textile demand, textile production output, employment of production workers in the textile industry, investment in the industry, and textile prices. These variables would be related to each other and to other variables (such as total national income, the Consumer Price Index, interest rates, etc.), through a set of linear or nonlinear equations." [Pind81. Emphasis added].

The multi-equation simulation model describes the type of model developed during this project. Obviously, with less than 30 equations, the MCCR software model is much simpler than the example described in the quotation. To put this into perspective, commercial models that describe portions of the U.S. economy typically contain from 700 equations (Chase Econometrics model) to 1450 equations (Townsend-Greenspan model) [Broo84].

Developing the model

The model began with a basic set of 31 equations agreed on during late 1985 (the "Minimum Model"). After combining some, and simplifying others, the model as programmed contains 24 equations which describe economic behavior adequately for the purpose.*

Constant technological change. Throughout the work we addressed the need for continuing improvement in the capabilities of all MCCR software development/support environments. An important vendor argument against use of any GFE/environment is that any required standard would inevitably result in degraded product quality or productivity, because of the unprecedented rapidity of technological change in the computing fields. The rapidity of change renders one year's standard the next year's obsolescence.

Project Management Plan. Technion conducted the research as seven tasks. In Task 1 we developed a detailed Management Plan for the research. Subsequent task descriptions follow.

Task 2. Prepare and validate an Econometric Model for implementing an Ada-based software development/ support environment in contractor and DOD organizations. Technion updated data by surveying vendors, users, and developers of environments that appeared promising.

Task 3. Automate the econometric model on the USAF standard Zenith Z-100 microcomputer. The resulting prototype automated model is implemented as an application of the Lotus 1-2-3 "spreadsheet" program, and is intended for use by commanders responsible for development and support of MCCR software.

Task 4. Develop the technical user Documentation on the contents of the automated model, and on how to use the automated model.

* Colonel Nidiffer developed additional refinements after the prototype computer program was completed.

Task 5. Using the econometric model, investigate effects of providing a standard environment to contractors, based on contractors' strategy and motivations. Task 5 considered business and economic areas that are not addressed in the minimum model actually programmed.

In this task Technion looked briefly at vendors' business concerns, such as market shares and vested interests in their own software tools, technology, and products. This area, in which contractors' interests may oppose those of the government, seems to hold rich possibilities for future research.

Task 6. Using the econometric model developed in Task 2, investigate effects of alternative control strategies, based on organizational constraints to successful implementation. Nine distinct control strategies are considered. They represent different combinations of voluntary [e.g., Jovial] and required [e.g., Navy's CMS-2] use of GFE environments by contractors, in both initial development and post deployment support contracts.

Task 7. Using the econometric model, select strategies and prepare cost/benefit tables for implementing environments under varied conditions. In carrying out this task, Technion conducted sensitivity analyses of the work done in Tasks 5 and 6, and selected feasible USAF strategies for implementing environments under varied conditions.

Types of environment

Houghton [Houg85] defines four different types of environments, classified according to the level of support by phase:

* **FRAMING** environments, which concentrate on the activities of **systems definition** and **software definition** that occur before programming can begin. Framing environments usually support only **one specific** methodology. Examples include: **SDS**, **DREAM** and **USE**.

* **PROGRAMMING** environments, which support only the later phases (**programming** and **testing**) of the software life cycle. Some environments of this type support incremental compilation, and allow programmers to execute program fragments. Examples include: **ALS**, **Arcturus**, **Rational's Ada compiler**, and **Smalltalk**.

* **GENERAL** environments, which support **all phases of the software life cycle**. They usually **support more than one programming language** and do not require users to follow one

specific methodology. They contain basic tools (editors and text processors) that support all phases of the life cycle, and they may have advanced tools for certain phases. Examples include: **TRW's Software Productivity System (SPS-1)**, Boeing's **ARGUS**, **UNIX**, and the French **"Platine"** system.

* **METHODOLOGY-SPECIFIC** life cycle environments. Provide automated support for all life cycle activities. **No automated environment of this type exists beyond the conceptual stage.** However, some methodologies do provide limited support for the entire life cycle.

As shown in Figure I-2, most of the development work has been done in the area of quadrant III, the GENERAL environment. In the present research, Technion obtained descriptive data for ten environments [Appendix A]. Because this is still an area in which development work is quite new, we expected that a maximum of about a dozen environments could be found (most in quadrant III). We attempted to obtain data for at least two environments in each quadrant.

Quadrant I	**Quadrant II**
FRAMING [Systems and Software Definition Phases] --	**PROGRAMMING** [Programming and Testing Phases] --
Examples: **DREAM, USE**	Examples: **ALS, Arcturus, Smalltalk**
METHODOLOGY-SPECIFIC [Automated Support for ALL Phases of System Life Cycle]	**GENERAL** [All Phases of SOFTWARE Life Cycle]
Examples: **NONE** (June, 1985)	Examples: **SPS, ARGUS, UNIX, Platine**
Quadrant IV	**Quadrant III**

Figure I-2. Types of Recent Software Environments.

<u>Leverage from Cost Drivers</u>

In looking for the few software cost drivers that USAF can manage, and **providing the control "levers" needed** to increase productivity, we assigned the following COCOMO factors to types of environment.

Quadrant I	**Quadrant II**
Framing [Definition Phase]	Programming and Test
(Product Complexity) CPLX (Required Reliability) RELY (Required Schedule) SCED (Volatility of Requirements) RVOL (Volatility of Target Computer) VIRT (Experience with Target Computer) VEXP	TURN (Host computer Response Time) MODP (Use of Modern Programming Practices) TOOL (Use of Powerful Software Tools) LEXP (Experience with Language Used) LIBR (Reuse of Software)
All tools useful in Quadrants I and II	All tools useful in Quadrants I and II.
Methodology-Specific	General [All Phases]
Quadrant IV	**Quadrant III**

Figure I-3. Regions of Greatest Leverage of Functions described by COCOMO Effort Multipliers.

USAF **can manage** six cost drivers in the systems and software definition phase (quadrant I, "Framing" environments): CPLX, RELY, SCED, RVOL, VIRT, and VEXP. SPO choices for the relative values described by these cost drivers have the greatest effect on ultimate project costs.

Similarly, in the programming and software test phase (quadrant II, "Programming" environments), USAF can manage five cost drivers: TURN, MODP, TOOL, LEXP, and LIBR. Typically these affect only a limited amount of total project cost.

USAF can manage all of these cost drivers in quadrant III ("General," supporting all project phases) and IV ("Methodology-specific," supporting all phases) environments. The relative importance of different phases, both in effort (work-months) and duration (months) are shown in Figure I-4; totals for work-months and duration sum to 100 percent. Although it consumes least effort of the phases in the COCOMO model, the Plans and Requirements phase influences effort needed in all subsequent phases.

PROJECT CHARACTERISTICS		PRIORITIZED COCOMO	SOFTWARE TOOLS
Work-Mos	Duration	Effort Multipliers	APPLICABLE
Plans & Requirements			
7%	15%	**ACAP**, CPLX, **AEXP**, RELY, TIME, DATA, VEXP, SCED, **TOOL**, **MODP** [5 E.M.s not relevant]	Analyst capability and experience might be increased by methodologies or expert systems techniques. Improving TOOL and MODP would have little effect in this stage.
Product Design			
17%	20%	CPLX, **PCAP**, **ACAP**, **AEXP**, RELY, TIME, STOR, VIRT, SCED, [6 E.M.s not relevant]	Capabilities and experience of Analysts and Programmers might be increased by methodologies or expert system techniques. Improving TOOL and MODP would have little effect in this stage.
Programming			
51%	40%	CPLX, **PCAP**, **ACAP**, **TURN**, **TOOL**, **RELY**, **MODP**, **TIME**, VIRT, **STOR**, **AEXP**, VEXP, LEXP, SCED. [DATA not relevant]	Bold-faced EM's can be improved by environments.
Integration and Test			
25%	25%	RELY, CPLX, **PCAP**, **MODP**, **ACAP**, **TOOL**, TIME, STOR, VIRT, VEXP, TURN, DATA, **AEXP**, **LEXP**, SCED	
100%	100%		

Figure I-4. Productivity Enhancement Leverage.

Hypothetical environments used

Five hypothetical environment configurations were defined in an earlier study [Werl85]. They were termed "HAPSE I" to "HAPSE V," from Hypothetical Ada-based Programming Support Environment.

The minimal groups of software tools [for HAPSE I] are:

Compiler/assembler
Linker/loader

Text formatter

Editor

Tool capabilities for subsequent HAPSE versions

Tool capabilities are added in evolutionary developments of the HAPSE. Figure I-5, "HAPSE Configurations," shows that HAPSE versions II to V contain the four minimal tools plus systematic additions to tool capabilities.

TOOL CAPABILITY	HAPSE II	HAPSE III	HAPSE IV	HAPSE V
Requirements tracing	X	X	X	X
Debugger	X	X	X	X
Cross-reference analyzer	X	X	X	X
Call structure analyzer	X	X	X	X
Configuration management	X	X	X	X
Standards auditor	X	X	X	X
On-line help	X	X	X	X
Design support		X	X	X
Statement coverage analyzer		X	X	X
Timer/performance analyzer		X	X	X
Project control		X	X	X
Syntax-directed editor		X	X	X
Menus		X	X	X
Requirements language			X	X
Graphics generator			X	X
MIL/SPEC generator			X	X
Typesetter			X	X
On-line documentation			X	X
Locked security controls				X

[Werl85], p. III-27

Figure I-5. "HAPSE" Configurations.

Benefit Factors Used

Three sets of benefit factors were used, roughly correspon-ding to projected tools in for HAPSE versions II, III, and IV.

COCOMO ATTRIBUTE	DESCRIPTION	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995
"HAPSE II" multipliers											
LIBR	Use of existing [reusable] software	0.89	0.88	0.88	0.88	0.87	0.87	0.87	0.86	0.86	0.86
TOOL	Use of software tools	1.01	1.01	1.01	1.00	0.99	0.99	0.99	0.98	0.97	0.97
TURN	Host Computer response time	1.01	1.00	1.00	1.00	0.99	0.99	0.99	0.99	0.98	0.98
MODP	Modern programming practices	0.95	0.94	0.94	0.93	0.93	0.92	0.92	0.92	0.91	0.91
LEXP	Experience with language	1.02	1.02	1.02	1.02	1.01	1.01	1.01	1.01	1.01	1.00
SEN	Experience with environment	1.03	1.02	1.02	1.02	1.01	1.01	1.01	1.00	1.00	1.00
"HAPSE III" multipliers											
LIBR	Use of existing [reusable] software	0.86	0.82	0.79	0.74	0.71	0.67	0.62	0.58	0.55	0.51
TOOL	Use of software tools	1.00	0.98	0.97	0.95	0.93	0.92	0.90	0.88	0.87	0.85
TURN	Host Computer response time	1.00	0.99	0.98	0.97	0.95	0.95	0.93	0.92	0.91	0.90
MODP	Modern programming practices	0.94	0.93	0.93	0.92	0.90	0.89	0.88	0.87	0.86	0.85
LEXP	Experience with language	1.02	1.01	1.01	1.01	1.01	1.00	1.00	0.99	0.99	0.99
SEN	Experience with environment	1.02	1.01	1.00	0.99	0.98	0.98	0.97	0.96	0.96	0.95
"HAPSE IV" multipliers											
LIBR	Use of existing [reusable] software	0.83	0.78	0.72	0.66	0.60	0.54	0.49	0.43	0.37	0.32
TOOL	Use of software tools	0.98	0.97	0.95	0.92	0.90	0.87	0.85	0.83	0.80	0.78
TURN	Host Computer response time	0.99	0.97	0.95	0.93	0.91	0.89	0.87	0.85	0.83	0.81
MODP	Modern programming practices	0.93	0.91	0.89	0.87	0.85	0.84	0.82	0.80	0.78	0.77
LEXP	Experience with language	1.01	1.00	0.99	0.98	0.97	0.96	0.95	0.94	0.93	0.92
SEN	Experience with environment	1.02	0.99	0.99	0.98	0.97	0.96	0.95	0.94	0.93	0.92

Measuring Costs and Benefits

The most difficult part of the modeling process is assuring that all relevant benefits and costs are included, somehow, in the equations. For example, many "economic" benefits do not appear in USAF reports. In the case of "indirect" costs and benefits, even when they can be identified they cannot necessarily be measured. One can see this in Table I-1 and in Table I-2.

Each table shows cost or benefit accounts (many of which are not reported). In Table I-1, the "++" symbol indicates benefit, and the "--" indicates loss. The "??" indicates an account for which it is not possible to predict net effect in advance. Note that in many cases, "benefits" to the Government are accompanied by "losses" to contractors. In both tables, **bold-faced** entries are particularly difficult to measure.

	BENEFITS TO GOVERNMENT			
	GOVERNMENT		CONTRACTOR	
BENEFIT ACCOUNT	Direct	Indirect	Direct	Indirect
Benefit over "Known Environment"	++	++	--	??
"Non-Proliferation" Costs	++	??	--	??
Avoidance of "Retooling"	++	??	++	++
Avoidance of "Contractual Lock-In"	++	++	--	--
Lower costs to produce and support MCCR software	++	++	??	??
Less reliance on programmers' skills	??	??	++	??

Table I-1. Summary of Tangible Benefits in Econometric Equations.

The same phenomenon is true for the cost side. Table I-2 lists some of the economic costs that USAF pays. The "++" symbol shows types of cost that are incurred and can be traced in accounting records. The "??" symbol represents costs that may not be traceable. In general, indirect costs, which must be estimated and allocated, present the greatest problem.

ACCOUNT	COSTS TO GOVERNMENT			
	GOVERNMENT		CONTRACTOR	
	Direct	Indirect	Direct	Indirect
Capital Costs				
Acquisition of GFE Environments (H/W, S/W, etc.)	++	++	??	??
Site Preparation	++		??	??
Version upgrades	++			??
Data Comm, Security	++	??	++	++
Depreciation of Environments, etc.		??		++
Operating Costs				
Maintenance of Environments and related H/W,S/W, etc.	++	++	++	??
Labor of USERS of Environments	++		++	??
Labor of Control Strategy Implementation (and defense) Staff	++	??	++	??
Labor of Program Management, Config. Mgt, and Procurement Staff	++	??	++	??
Cost of Program Delays associated with control strategy	++	??	??	??
Fringes on Labor cost	++	++	++	++
Rent, light, power, supervision, etc.	++			++
Training of Users	++		++	??

Table I-2. Summary of Tangible Costs in Econometric Equations.

Replacing labor costs with investments in environments.

USAF s cost for direct labor (in the form of USAF "spaces," "faces" or "contract dollars") really dominates the economic analysis. They provide the tangible leverage USAF can exchange for the many virtues of a standard environment. This is true simply because the current cost for labor in the U.S. is four or five times the cost for the environments and software tools.

USAF probably spends little more than $10,000 average initial cost and $2,000 per year variable cost for each of its present environments. That is about $12,000, or about an eighth of USAF cost for direct labor. That expectation justifies our assumption that environments will produce tangible economic benefits by replacing some of the labor cost now projected to be needed in five years.

This is not to say this approach will reduce labor costs. We know that is not feasible, considering the many political forces at work. Instead, we are suggesting a way to reduce the rate of exponential growth -- i.e., to keep those costs from continuing their incessant eating away at funds that might be better applied to other equipment/services needed by USAF, as they have for the past decades.

Accomplishment

Task 2 - Preparation of econometric model. We completed preparation and validation of the econometric model for implementing an Ada-based software support environment in contractor and DoD organizations. The equations were developed to describe decision-making as it takes place within AFSC.

In developing the model equations, we:

a. Selected data and transformed them into consistent variables.

b. Analyzed data collected, using statistical methods, and developed econometric formulas describing benefit/cost behavior of the variables. Together, we selected from the many feasible equations, a set of equations that is usable and that provides for the greatest practical predictive ability.

c. Equations were augmented by non-quantitative decision tables that describe alternative organizational behaviors, control strategies, and policy directives.

d. We were not able to complete the set of equations that describe "off-books" costs and benefits such as the effects of depreciation/amortization, and "opportunity costs" that cannot be ignored by contractors.

Clearly, this was a challenging task. It could only be considered because the underlying research had been done. The "COCOMO" project database, which includes 63 software projects completed during the years 1964-1979, provided consistent descriptions for projects done during a 15-year period characterized by rapid evolution in the practice of software development. Of the 63 projects listed, 34 were software projects similar to USAF mission-critical software projects.

However, the COCOMO project database includes no projects that used the Ada language or integrated environments; none were available during the time covered. The COCOMO project database needed to be augmented with data on more recent projects. Our efforts were not effective in meeting these needs. We were surprised to find, for example, that data from the NASA Software Engineering Laboratory's research and from RADC were of little help. This is because many data elements in those data bases were not collected, or were recorded using different definitions.

We obtained updated development and "maintenance" data for COCOMO Effort Multipliers: MODP, TOOL, TURN, LEXP. We extended present effort multipliers to estimate effects of using of software tools integrated into one software environment ("SEN"), and for reuse of standard code fragments ("LIBR").

We gave special attention to six variables, known to directly improve productivity, by selecting software tools for inclusion in an integrated software support environment. The individual variables include:

(a) An integrated set of automated software tools

(b) Modern programming practices [COCOMO "MODP"].

(c) Higher order languages (such as Ada)

(d) Use of existing software, from libraries of "reusable code fragments. We defined and arbitrarily assigned the acronym "LIBR" to this variable.

(e) Hardware and software that provides response time short enough and memory capacity large enough to avoid interfering with programmers' trains of thought. [COCOMO "TURN"].

(f) Design of tools, such as syntax-directed editors, that help users speed improvement in learning (e.g., to gain experience with language(s) used, type of software application, and the "virtual" hardware/software/procedure machine for which the software is produced).

Ray Houghton obtained data for the econometric model from telephone surveys, government sources, and from published literature. He contacted government and industry users, vendors, and literature to obtain: (1) percentage of their effort that is plowed back into R&D; (2) percentage of resources and staff assigned to support the environments; (3) percentage of resources and staff required to obtain incremental improvements in performance; and (4) length of environment "version life." Examples of the environment data are given in Appendix A. Data were organized by environment type, and (where available -- only rarely) include cost of developing each environment, time to complete development of each environment, size (in lines of source and/or object code), and average version life in years. We attempted to obtain a minimum of two samples of data for each of the four environment types (shown in figures 3 and 4 of the December Management Plan). Colonel Nidiffer obtained current project support data from AFLC installations and from Tinker AFB support operations.

Task 2 - Model Validation. We were not able to obtain valid data on historical USAF software development projects to use in calibrating the selected equations. This unexpected situation caused us real difficulties in the tasks of validating, "tuning" and calibrating the model's equations.

Task 3. The Technion/SASC team automated the econometric model developed in Task 2, on the USAF standard microcomputer (Zenith Z-100). The model was programmed as an application of the LOTUS 1-2-3 system (version 1.5), a "spreadsheet" program widely used within USAF. On June 20, the resulting code was delivered on a floppy disk (CDRL sequence 7).

The automated model we delivered is a prototype. As we cautioned in our Management Plan:

> "3.3.1 The Technion/SASC Technologies team will exercise great care to ensure that the delivered product meets the specifications. Nevertheless, we must be mindful of past experience. When there is little precedent for use of a new technique [such as this evaluation model] in the field, and when particular applications have not been programmed before, initial versions of software products are often closer in nature to prototypes than to seasoned "products. In the present project for example, the initial model will include only the econometric equations and data elements determined in Task 2. Recognizing this, we must caution that -- in spite of all our care -- the delivered product might be seen by some as not a fully validated and bug-free tool totally adequate for operational deployment and use."

Mr. William P. Byerly, of SASC Technologies, Inc., programmed the application and wrote the user documentation [CDRLs 6 (User's Manual), 8, 9, and 10].

Tasks 5-7. Using the automated program, we made analyses. These involved making many runs, varying one factor at a time.

Four of the more significant results found are:

1. The model works.

2. A long time, at least four years, is required for USAF to achieve the benefits of an integrated automated software support environment.

3. A high "front-end" investment, ranging from $75 million to more than $200 million, is required.

4. USAF needs to put a systematic data collection program into place, in order to get meaningful data for evaluation efforts such as this.

Research results are described in more detail in Chapter Two.

CHAPTER TWO

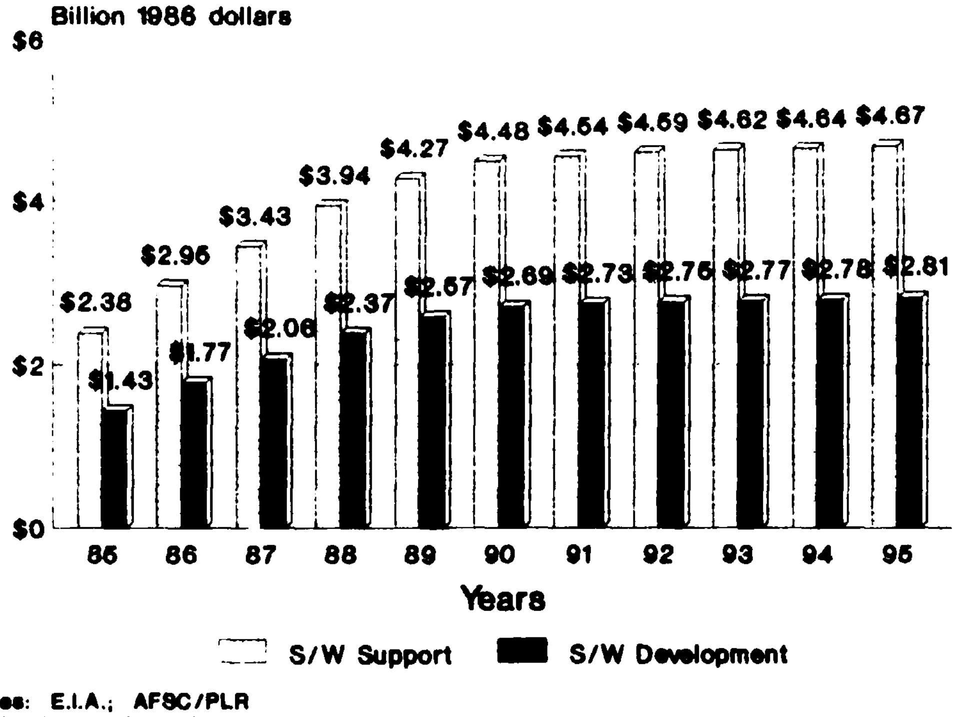
FINDINGS

Three categories of findings are presented. The first, "General," relate to the potential impact on USAF MCCR software. The second, "Characteristics," shows important properties of the model. The third,"Development of econometric model," describes findings regarding assumptions included in the model.

GENERAL FINDINGS

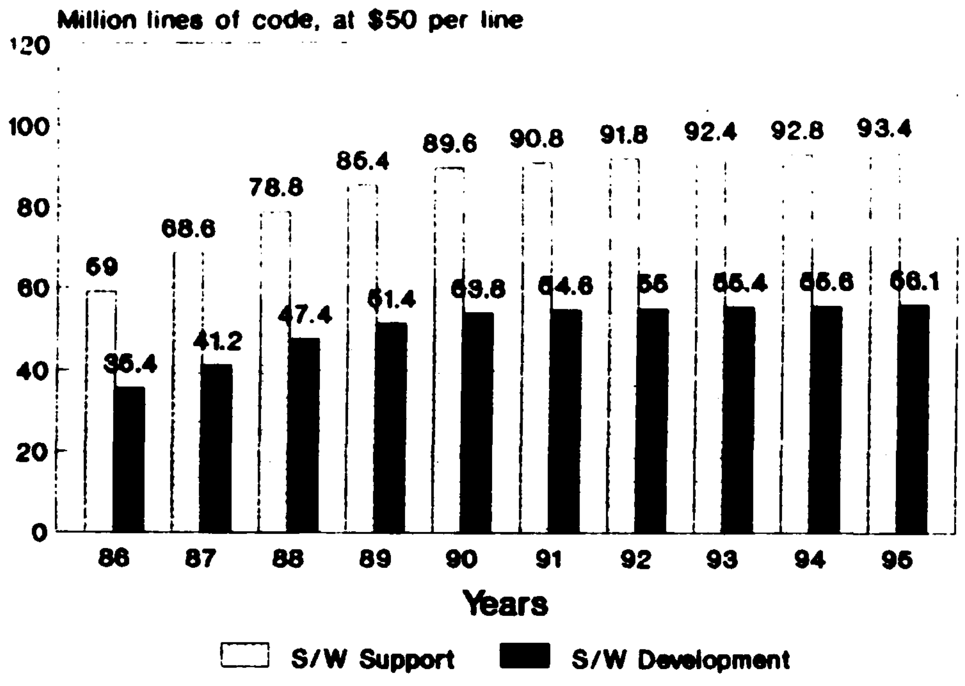
MCCR Software: Big and expensive business

If aircraft and missiles are the heart of USAF, MCCR software is the brain. It is expensive. To keep its present MCCR software operational and to develop new MCCR software, USAF will spend $4.7 billion in FY1987. By FY 1995 this will rise to $7.4 billion (1986 dollars). Figure II-1 pictures this growth, while Figure II-2 shows the size of USAF's growing inventory of MCCR software. Figure II-3 translates this into thousand work-years.



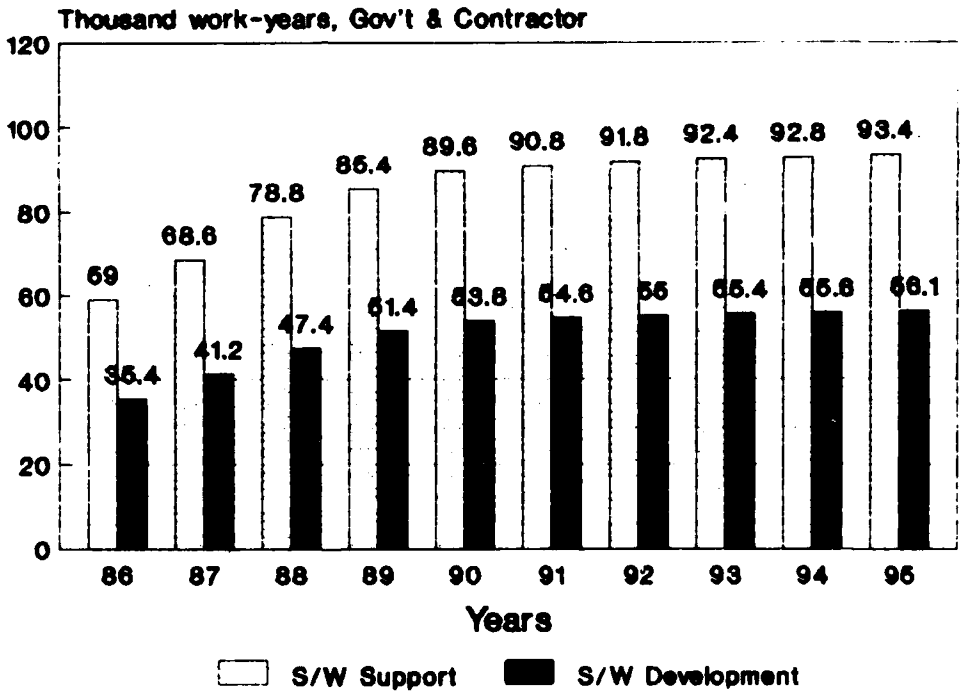
Sources: E.I.A.; AFSC/PLR
Technion International, Inc.

Figure II-1. USAF MCCR Software Costs.



Sources: E.I.A.; AFSC/PLR; and Technion International, Inc.

Figure II-2. MCCR Software Inventory is Large and Growing.



Source: Technion International, Inc.

Figure II-3. Assumed workload.

Table II-1 shows the assumptions we used in developing these figures, and others shown below.

Table II-1. Assumptions Used in Calculations.

1. ELECTRONICS INDUSTRY ASSOCIATION'S 1986 ESTIMATE OF USAF MCCR SOFTWARE INVENTORY COST (Billion 1986 dollars)

	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995
Raw Estimate Data										
Software Support	2.95	3.60	4.35	4.95	5.45	5.80	6.15	6.50	6.85	7.25
S/W Development	1.77	2.16	2.61	2.97	3.27	3.48	3.69	3.90	4.11	4.35
Total Software	4.72	5.76	6.96	7.92	8.72	9.28	9.84	10.40	10.96	11.60

2. DEFLATION FACTORS AND ESTIMATE IN CONSTANT (1986) DOLLARS

	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995
Deflator (5.00%)	1.00	1.05	1.10	1.16	1.22	1.34	1.34	1.41	1.48	1.55
Est. Constant Dollars										
Software Support	2.95	3.43	3.94	4.27	4.48	4.54	4.59	4.62	4.64	4.67
S/W Development	1.77	2.06	2.37	2.57	2.69	2.73	2.75	2.77	2.78	2.81
Total Software	4.72	5.49	6.31	6.84	7.17	7.27	7.34	7.39	7.42	7.48

3. SIZE OF USAF MCCR SOFTWARE INVENTORY WORKED ON EACH YEAR** (In million delivered source instructions [DSI], at $50/DSI)

	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995
Software Support	59.0	68.6	78.8	85.4	89.6	90.8	91.8	92.4	92.8	93.4
S/W Development	35.4	41.2	47.4	51.4	53.8	54.6	55.0	55.4	55.6	56.1
Total Software	94.4	109.8	126.2	136.8	143.4	145.4	146.8	147.8	148.4	149.5

4. USAF WORKLOAD, MCCR SOFTWARE DEVELOPMENT AND SUPPORT* (Thousand work-years, at 1000 instructions per work-year)

	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995
Software Support	59.0	68.6	78.8	85.4	89.6	90.8	91.8	92.4	92.8	93.4
S/W Development	35.4	41.2	47.4	51.4	53.8	54.6	55.0	55.4	55.6	56.1
TOTAL	94.4	109.8	126.2	136.8	143.4	145.4	146.8	147.8	148.4	149.5

* According to an informal communication with AFSC/PLR, the total work-years (both USAF and contractor) for calendar 1986 is in the neighborhood of 120,000. This estimate, therefore, is in the ballpark. Without more detailed knowledge, these estimates cannot be refined further.

** Total inventory may be as great as ten times the amount worked on during any one year. Accuracy of the estimate, however, is subject to great uncertainties.

CHARACTERISTICS

Two control strategies offer highest net benefit

In controlling use of the integrated automated environment, two strategies offer the highest net benefits. All others yield negative benefits.

Both of the strategies approximate the control strategy USAF has used for the JOVIAL language. The strategies are:

1. Mandatory use for post-deployment support, but no extraordinary requirements during development. This implies that development contractors can use tools and techniques of their choice, subject to rigorous acceptance testing by USAF to assure conformance to applicable standards.

2. Mandatory use for post-deployment support, with voluntary use during development. In effect, this is the same as strategy (1) above.

The relative net benefits of the nine strategies are shown in Figure II-4.

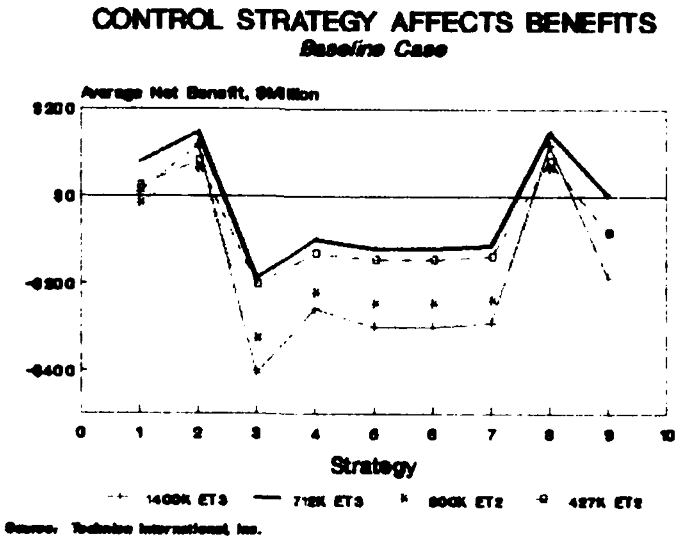


Figure II-4. Control Strategy Affects Net Benefits.

The nine strategies used are shown in Figure II-5.

Control Strategy	Required for use in: Development	Support
1	Mandatory	Mandatory
2	N/A	Mandatory
3	Mandatory	N/A
4	N/A	N/A
5	Voluntary	Voluntary
6	N/A	Voluntary
7	Voluntary	N/A
8	Voluntary	Mandatory
9	Mandatory	Voluntary

Figure II-5. Control strategies.

Costs lead benefits

Figure II-6 shows the time relationship for one example of mandatory use during development. Costs, which are incurred from

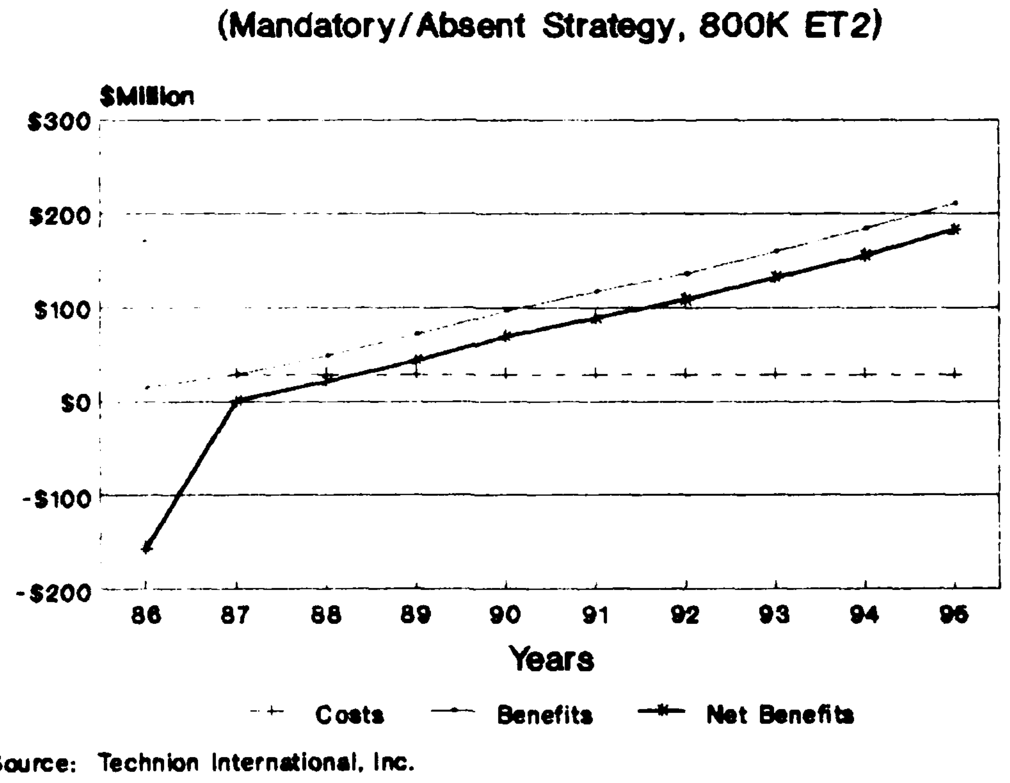
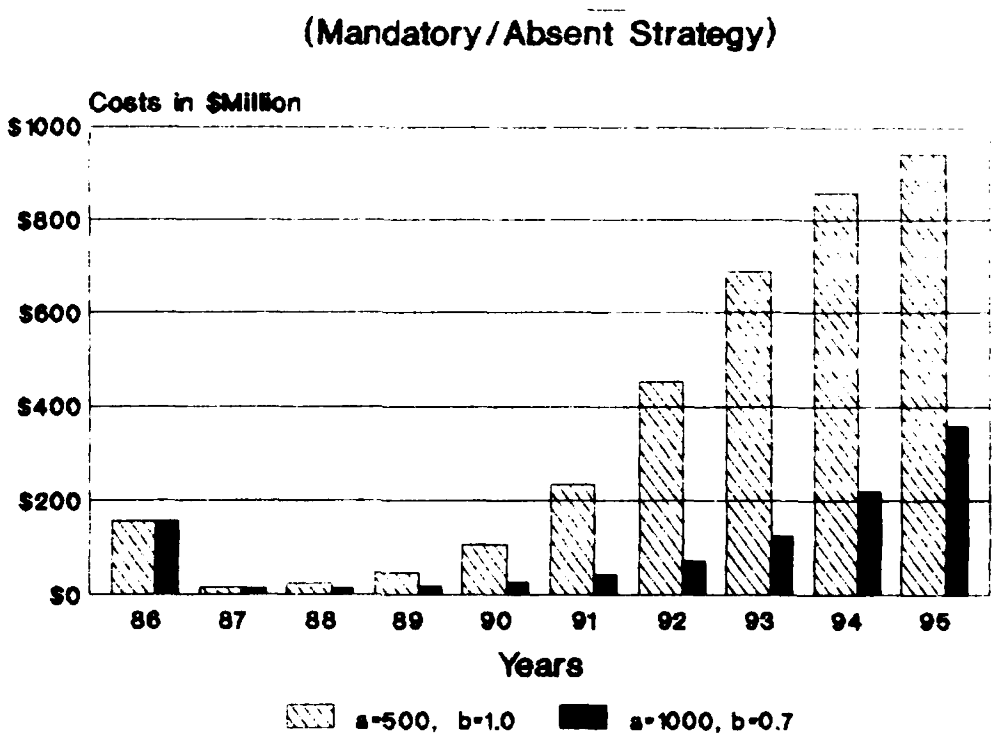


Figure II-6. Costs lead benefits.

the beginning of the program, precede the initial benefits by a year. Costs are then regained within a period of years, but the period may be impracticably long to produce a viable net benefit to USAF.

Rate of introduction has great leverage

The rate of introduction of standard environments exerts substantial leverage in determining the magnitude of the net benefits. Rapid introduction, so that the market is "saturated" [i.e., essentially universal] within two years, yields the greatest net benefit. Precedent for this can be found in the market trajectory of the IBM PC. Slow introduction, extending over four or five years, has least net benefit for USAF.
As shown in Figure II-7, USAF annual costs are higher [benefits held constant] for the slow introduction strategy.



Source: Technion International, Inc.

Figure II-7. Saturation Rate Affects Costs.

Current productivity rate affects benefits

Net benefits to USAF are greater when existing productivity is low, and less significant when productivity is already high. This is shown in Figure II-8. The average annual production rate, in lines of code per work-year (PLOC), is shown for three levels -- 3000, 2000, and 1500. The net benefit is clearly much higher for the low current levels of productivity.

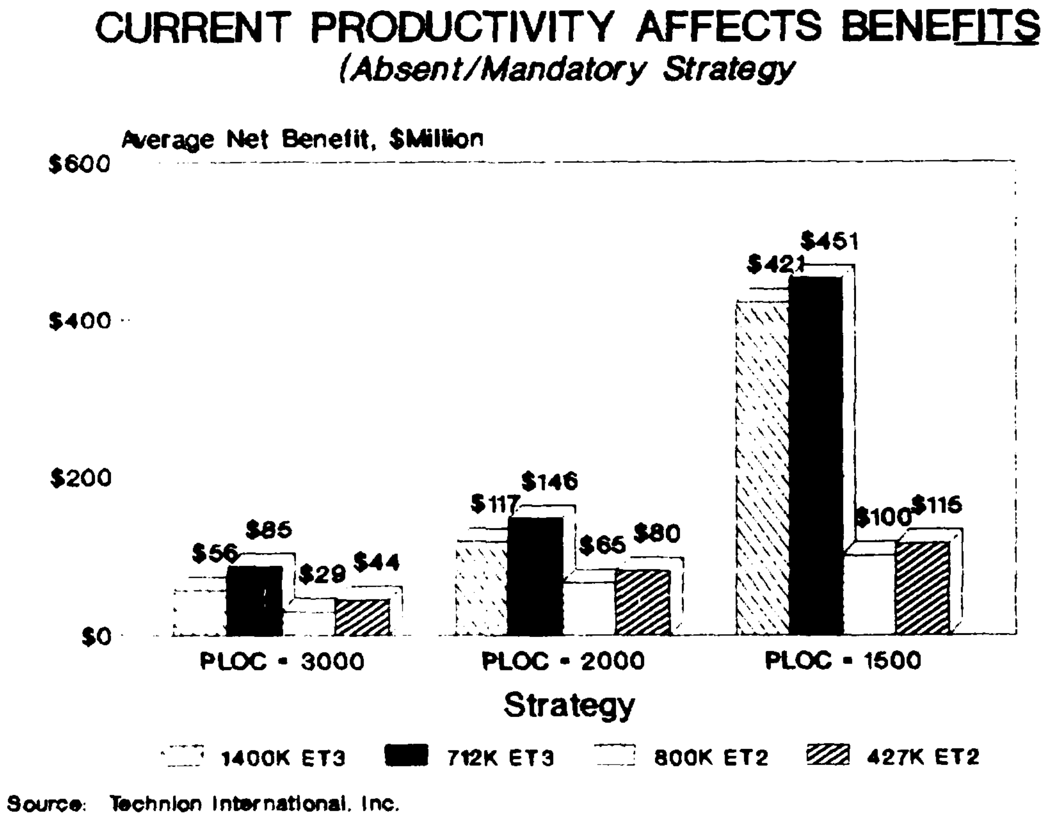


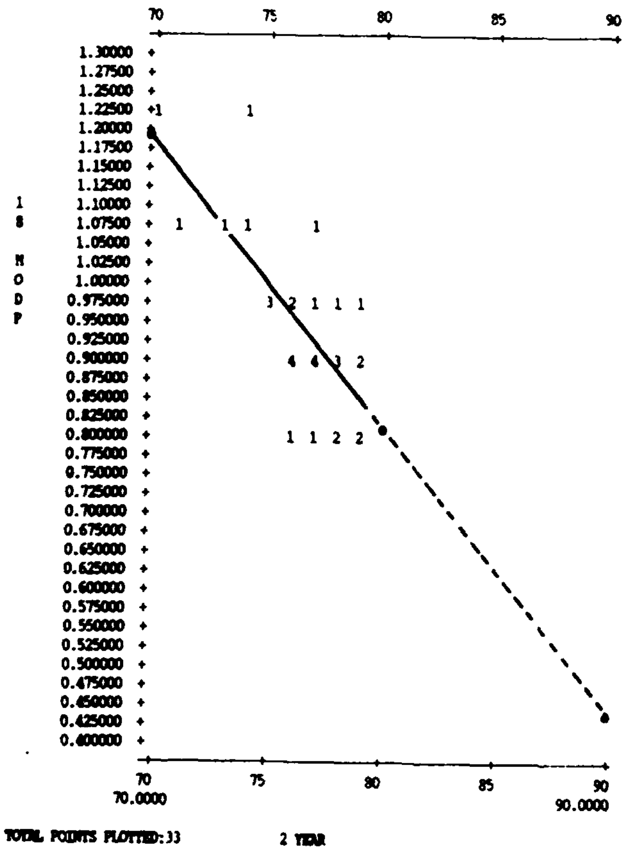
Figure II-8. Current productivity rate affects benefits.

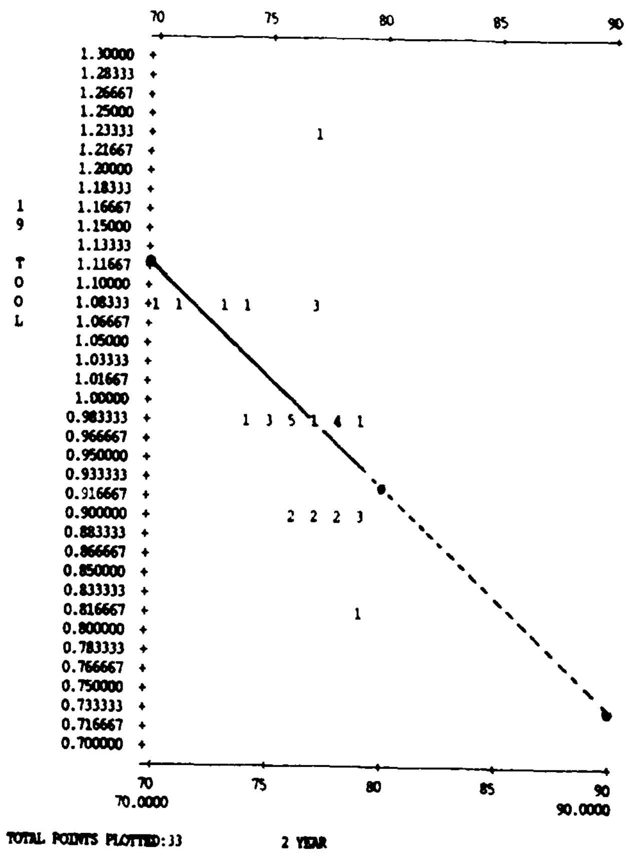
DEVELOPMENT OF ECONOMETRIC MODEL

The benefits computed by the model are derived from six cost drivers. Four of these (MODP, TOOL, LEXP, and TURN) are from the COCOMO model. Two others were derived from literature. They are: (1) SEN, "skill in using an integrated environment;" and (2) LIBR," reuse of existing proven library code fragments."

Statistical analysis

The four COCOMO drivers were derived from statistical analysis of 33 MCCR-like projects included in the 63-project COCOMO database [Boeh81, pp. 496-97]. Plots against time for two, MODP and TOOL, are shown in Figure II-4 along with the coefficients of correlation, R, and significant test values of the F-statistic. The numbers on the plots refer to number of observations at that point. Trend lines are least-squares lines for 1970-79, and are extended to 1990.





MODP

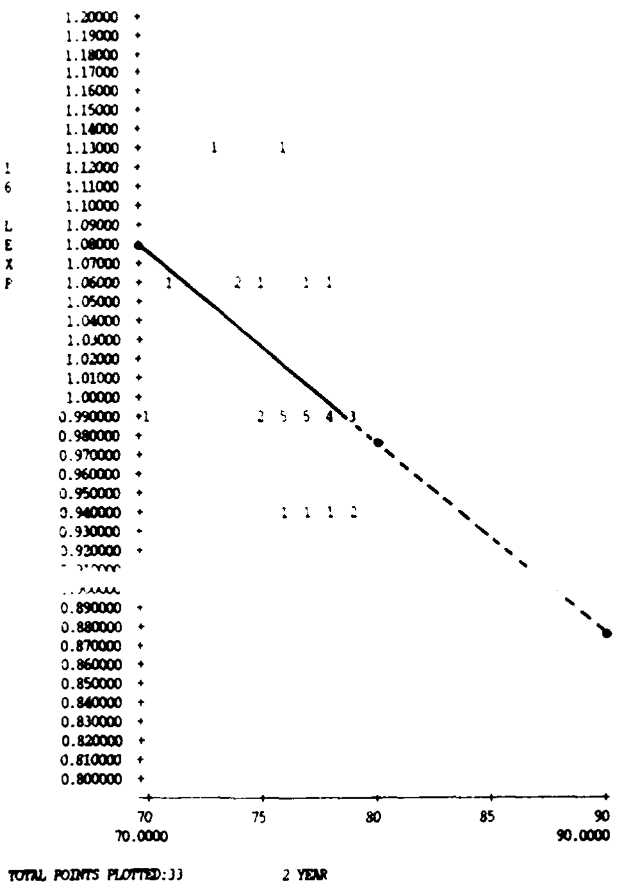
R = 0.733
F = 36

TOOL

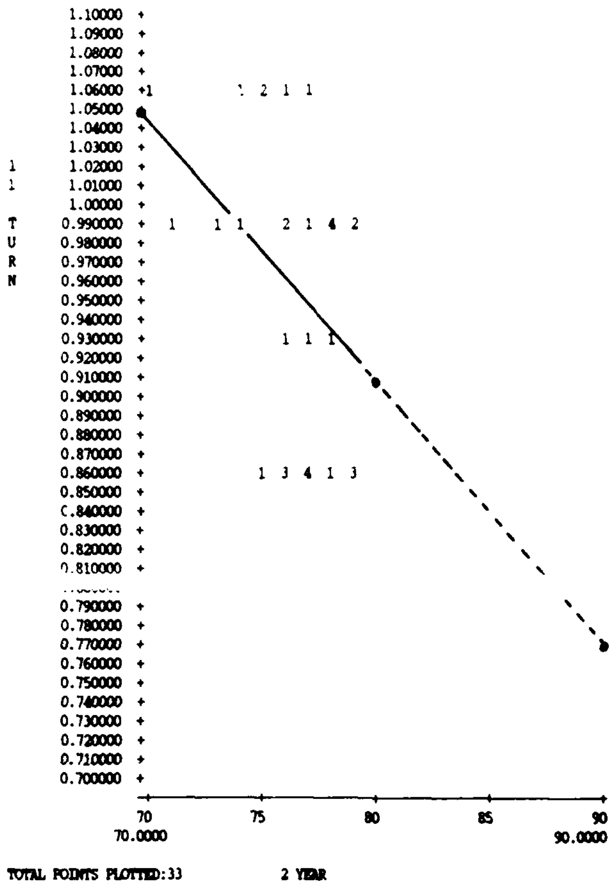
R = 0.49
F = 9.9

Figure II-9. Statistical plots of MODP and TOOL.

Similar plots for LEXP and TURN appear below, in Figure II-10.


LEXP

R = 0.456
F = 8.12

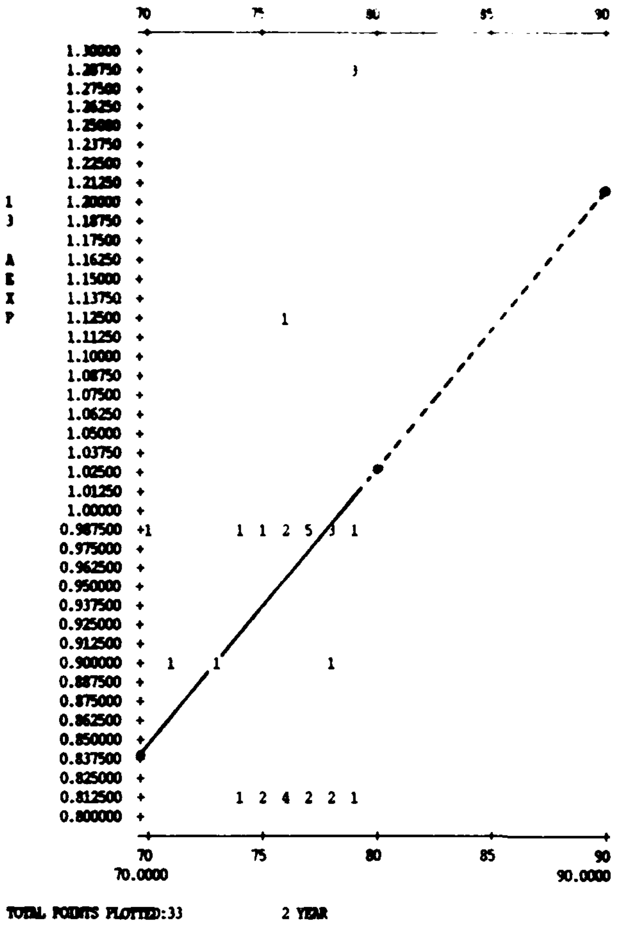

TURN

R = 0.393
F = 5.65

Figure II-10. Statistical plots of LEXP and TURN.

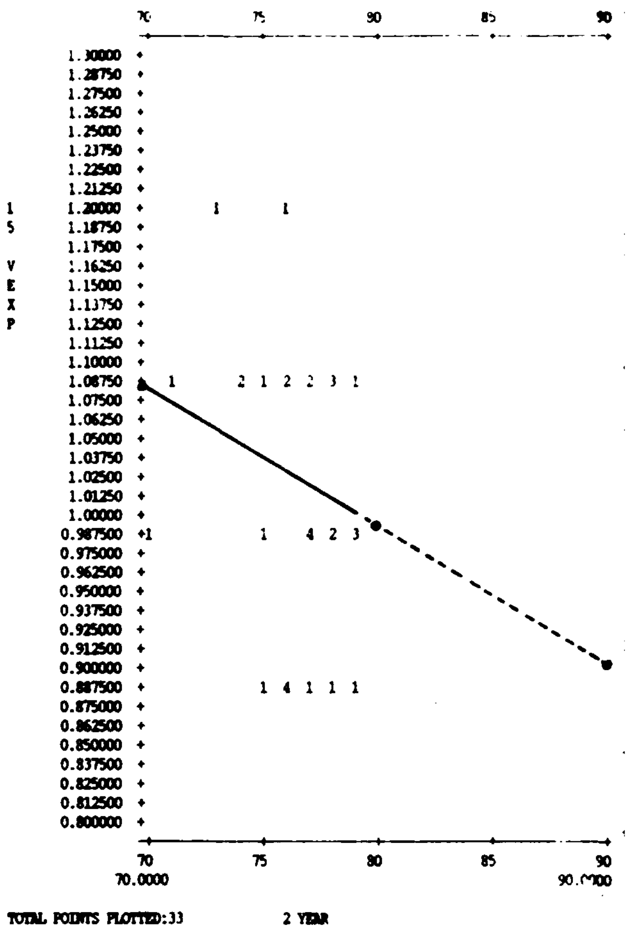
To estimate the SEN driver, we analyzed two related drivers. AEXP relates to the experience of the analysts, and VEXP relates to the team's experience with the target "virtual machine" [i.e., the target computer hardware, operating system, sensors, etc.]. Each of these drivers has a low level of significance, and their trends are in opposite directions. We exercised considerable judgment in interpreting both drivers. The plots are shown in Figure II-11.

For the effect of reusable code, we relied on projections by Boehm [Boeh81, p. 686], [Boeh84,p. 33].



AEXP

R = 0.290
F = 2.84



VEXP

R = 0.212
F = 1.46

Figure II-11. Statistical plots of AEXP and VEXP.

CHAPTER THREE

CONCLUSIONS

In this chapter we present a summary of conclusions from all areas studied during the project. We follow the summary with our recommendations for future modifications to the model.

SUMMARY OF CONCLUSIONS

Three categories of conclusions are presented. The first, "General," relate to the potential impact on USAF MCCR software. The second, "Econometric model," describes strategies for controlling use of a GFE/environment, supported by results of exercising the model. The third, "Characteristics," summarizes the important properties of the model.

General

Results of the research make it clear that:

1. The annual costs of maintaining the existing inventory of non-standard MCCR computer programs are immense. Costs projected by EIA are $7.17 billion for FY 1990 [EIA86].

2. No systematic measures to improve productivity for acquisition and support of MCCR software are currently in place. For the "business as usual" option - without significant USAF action - total costs to USAF may be greater.

3. It is feasible for USAF to acquire integrated automated software support environments. Perhaps a dozen U.S. vendors can build an integrated, automated environment for Ada MCCR software with today's technology. Such environments could harness known software engineering technology to produce annual improvements in productivity of more than ten percent. Some contractors have observed such annual increases in productivity in their internal operations [Boeh81], [Boeh82], [Boeh84], [Bita85], and [Werl86].

   Technion International recommends that USAF undertake such an effort, learning from the recent experience with such related programs as the Army's ALS compiler.

4. By itself, this recommendation takes too much time. A minimum of four years (and $50 million) is needed to field a "HAPSE II level" environment, capable of helping USAF improve MCCR software productivity by about six percent annually after delivery. The "HAPSE V level" environment, which would help improve productivity by about ten percent annually after delivery, would need an additional three years (and $150 million).

5. USAF can begin improving productivity in development of MCCR software without waiting for additional software tools. Improved productivity is aided by, but is not dependent on, availability of an improved software development environment. Systematic productivity improvement efforts are well understood and clearly successful. They rely on better use of existing tools, on careful determination of requirements, and on reuse of existing software "code fragments." [Boeh81, pp. 641-89].

Econometric Model

1. In controlling use of the integrated automated environment, two strategies offer the highest ret benefits:

   (a) Mandatory use for post-deployment support; with no extraordinary requirements during development.

   (b) Mandatory use for post-deployment support; with voluntary use in development.

   Both of these options approximate the control strategy USAF has used for the JOVIAL language.

   For all other options, benefits are negative.

2. The rate of introduction of standard environments has great leverage over the magnitude of net benefits. Rapid introduction, so that use is widespread within two years, has greatest net benefit. Slow introduction, extending over four or five years, has least net benefit for USAF.

3. Net benefits to USAF are greater when existing productivity is low, less significant when productivity is high. That is, operations with the lowest productivity would benefit most from using an integrated automated software environment.

Characteristics simulated by the model

The model can be used to simulate behavior by varying control strategies, environment characteristics, profiles for cost and workload (development and support) over a ten-year period, and model parameters. These changes are made by choosing "EDIT" from the main menu, then by selecting "BENEFITS," "STRATEGY," "ENV-TYPE," "YEAR," or "CONSTANTS." Details are given in the companion volume.*

To modify benefits

Benefits, controlled by the BENEFITS.WKS file, can be modified in three principle ways:

1. Year of introduction of the environment can be changed, by EDITing the file.

2. Rates of improvement can be altered, for the six effort multipliers now in the file, by changing the tabled annual multiplier values.

3. Other multipliers and formulas can be added to the ECONOMET.WKS file. A programmer knowledgeable in LOTUS 1-2-3 is needed for this option.

To modify strategy

Strategy can be modified in four ways. One can change:

1. The number of government sites using a strategy.

2. The number of contractor sites subject to a strategy.

3. The weighting factor, WSCj.

4. From the PARAMETR.WKS file, the average number of systems per site.

To modify an environment

An environment can be modified by editing the ENVDEF.WKS or the BENEFITS.WKS worksheets, and the PARAMETR.WKS table's "COST" option. Multiple characteristics and costs can be changed.

To modify costs

Costs can be changed by editing "PARAMETR.WKS" and choosing the "COST" option. Multiple costs can be changed.

---

* Byerly, William P. Automated Econometric Model User's Guide. June 20, 1986.

To change workload

The workload, in lines of code to be developed and supported, can be changed in the "COSTS" option of PARAMETR.WKS.

To modify constants

Rate of penetration of the "market" for environments, controlled by the "Pearla" and "Pearlb" parameters in ECONOMET.WKS, can be changed to simulate faster or slower acquisition and installation of environments at government and contractor facilities. This is done from the "CONSTANTS" menu selection.

The interest rate, used to compute present value of annual costs over the ten-year simulation period, can be changed from the present value of 0.1 [10.0 percent]. This is also done from the "CONSTANTS" menu selection.

RECOMMENDED MODIFICATIONS FOR SUBSEQUENT VERSIONS

We recommend that several enhancements be considered for future work.

1. Add more detail for interim steps in the process of computing costs. Substantial detail is provided for in the file design, but more definitive formulas are needed to use the capability. Additional printed detail should include:

    a. The number of environments to be acquired.

    b. Capital costs for acquisition and site preparation.

    c. Direct operations and maintenance costs, by category, such as training

    d. Add formulas using variables in the "Environment Type" file. The file is active but at present has no entries. More precise definition of contractor variables can be attained by activating this module.

2. Change sequence of menu steps. Many hours of exercising the model have shown that menu steps, particularly for selecting strategies and environments and for editing, can profitably be resequenced to reduce required keystrokes.

3. Activate the "Environment Type" module.

4. Add cumulative return on investment (ROI) by year.

CHAPTER FOUR

RECOMMENDATIONS FOR IMPLEMENTATION

Recommendations are in two parts, those dealing with organizations external to USAF and those inside.

External

1. Consult with appropriate Congressional committees for acceptable ways to implement these findings. An approach like that which led to the "Warner Amendment" would probably be most successful. House Government Operations Committee members have traditionally opposed any measure that seems to interfere with "full and free competition," and have challenged DoD initiatives for standardization.

2. In contrast, the Armed Services committees would be most likely to support USAF and DoD action to standardize software support environments. With this external support, internal USAF actions can be successful. Without it, we can expect to have new layers added to the existing "scar tissue" at DoD.

Internal

1. Technion International recommends that USAF begin a systematic three-pronged effort to improve the state of MCCR soft-ware practice, learning from the recent experience with such related programs as the Army's ALS compiler. This approach is directed at improving the practice of MCCR software development and support, supplementing the many present efforts to further develop software engineering theory. Both are needed, but at this moment we believe USAF needs improvement in applied practice are more urgent.

(a) Competitively acquire three automated integrated software environments to develop and support MCCR software. Perhaps a dozen U.S. vendors can do this today (e.g., General Electric, TRW, Softech, Microsoft). Such environments could harness known software engineering technology to produce annual improvements in productivity of more than ten percent. [See Appendix D].

By itself, this approach takes too much time. It could be shortened if vendors reuse existing code. Starting from scratch, it would take a minimum of four years (and $50 million) to field a "first level" environment, capable of helping USAF improve MCCR software productivity by about six percent annually after delivery. The "third level" environment, which would help improve productivity by about ten percent annually after delivery, might take an additional three years (and $150 million).

(b) Begin a systematic productivity improvement effort program inside USAF. The technology is well understood and clearly successful. It relies on better use of existing tools and on reuse of existing software "code fragments." USAF can begin improving productivity in development and support of MCCR software immediately, without waiting for additional software tools. Improved productivity is aided by, but is <u>not dependent on</u>, availability of an improved software development environment.

(c) Competitively acquire three libraries of Ada language MCCR software "fragments," for systematic reuse. This requires two efforts by each vendor: (1) test and validate candi-date existing code fragments; and (2) classify and catalog them in ways that permit users to quickly identify promising candidates for reuse.

BIBLIOGRAPHY

[Alex86] Alexandridis, Nikitas A. "Adaptable Software and Hardware: Problems and Solutions." IEEE Computer, Vol. 19, Nr. 2, pp. 29-39.

Paper surveys MCCR software problems and literature and concludes that adaptable, reusable software and hardware are essential to mission-critical computing systems.

[Arth83] Arthur, Lowell Jay. Programmer Productivity: Myths, Methods and Murphology. New York: John Wiley. 1983.

Book describes methods for improving software productivity used by a unit of American Telephone and Telegraph company. Reusable code is one element in the program described.

[Baro86] Baroudi, Jack J., and Ginzberg, Michael J. "Impact of the Technological Environment on Programmer/Analyst Job Outcomes." Communications of the ACM. Vol. 29, No. 6, June 1986, pp. 546- 555.

Report of survey that attempted to relate technological factors in analyst/programming jobs. The results are useful as an early example of the type of research that needs to be done in this field. However, only 11 percent of the variance in job satisfaction [not a high proportion] is explained by the variables studied.

[Bars84] Barstow, D. R., Shrobe, Howard E., and Sandewall, Erik. Interactive Programming Environments. New York: McGraw-Hill Inc. 1984.

Book is a collection of papers by authorities in the field of interactive programming environments. It present developments to about 1982 from the fields of programming methodology, artificial intelligence and software engineering. The book describes how to save time and increase productivity by using interactive programming environments.

[Bask86] Baskette, Jerry. "Life Cycle Analysis of the AIM Project." ACM Special Interest Group on Ada, Ada Letters, Vol. VI, Nr. 2, March/April 1986, pp. vi.2-86 to vi.2-90.

Paper presents comparison of life cycle effort (work-months) and durations (months) observed on the "AIM" project (which was not further defined). Compares percentages to those of the Softcost and COCOMO models, and to the proprietary GTE model.

During the AIM project a heavy emphasis was laid on requirements, specification and design. Authors thought this had reduced the number of errors and the required testing and debugging time. No quality data were presented.

[Beau86] Beauchamp, Marc, with Katayama, Hiroko. "A few ugly facts." Forbes. August 25, 1986, pp. 100-104.

Article cites "a few ugly facts" that challenge the conventional wisdom about Japanese in software. "When the Japanese began setting their software sweatshops, U.S. programmers sneered -- computerdom's equivalent, they said, of writing Shakespeare by committee. No one is sneering now. What counts in software these days, argues George Lindamood, a Washington D.C.-based computer consultant who worked for Burroughs in Japan, 'isn't creativity and innovation as much as delivery on time and budget'." Also see [Bela86] and [BusW84].

[Bela86] Belady, Laszlo A. "The Japanese and software: is it a good match?" IEEE Computer. Vol. 19, No. 6, June 1986, pp. 57-61.

Paper is a personal account of the author's 18 months in Tokyo, working in the IBM Japan Science Institute, helping to develop a software technology research group. He found that "the real down-to-earth software technology effort resides in . . . six companies [in Japan]. This effort is aimed at improving productivity and quality in an industrial every-day sense. The companies are [in Belady's sequence]: Nippon Electric, Hitachi, Fujitsu, Toshiba, Oki, and Mitsubishi.

He concluded that the best "counterstrategy for Japan's competitors [including those in the U.S.] may be to become a moving target. The moving target--the product of [U.S.] flexibility--is very difficult to follow, particularly if you are already following a very good plan [as the Japanese tend to do]. Also see [BusW84] and [Beau86].

[Bita85] Bitar, Imad, Penedo, Maria H., and Stuckle, E. Don. "Lessons Learned in Building the TRW Software Productivity System." IEEE paper CH2135-2/85/0/0350$01.00.

Authors describe TRW's "Software Productivity System," as it was developed from 1981 to 1984. TRW's productivity goals are ". . . to increase TRW projects' productivity by a factor of 2 in 1985 and by a factor of 4 in 1990, using 1980 as the baseline." Authors give examples of software tools used in the SPS. Important conclusions are: (a) development environments are mandatory for companies that develop large software systems since they provide a large payoff in productivity; (b) the man-machine interface must accommodate all classes of users and be consistent across tools; (c) sometimes increase in productivity cannot be measured so easily since it may mean increase in quality, not necessarily in quantity; and (d) building a software development environment involves initiatives in many areas and corporate commitments. Also see [Boeh82] and [Boeh84].

[Boeh81] Boehm, Barry W. Software Engineering Economics. Englewood Cliffs: Prentice-Hall. 1981.

Author presents an encyclopedic and highly detailed description of the "COCOMO" software cost estimating system. He devotes an entire chapter to a systematic description of methods for developing a software productivity improvement program. The book includes a consistent 63-project data base, from which the COCOMO equations were derived. Thirty four of the projects are broadly similar to USAF MCCR software projects.

[Boeh82] Boehm, Barry W., Elwell, James F., Pyster, Arthur B., Stuckle, E. Donald, and Williams, Robert D. "The TRW Software Productivity System." IEEE paper 0270-5257/82/0/0137$0.75.

Paper is an overview of the TRW Software Productivity System (SPS), an integrated software support environment based on the Unix(tm) operating system, a wide range of TRW software tools, and a wideband local network. Important conclusions are: (a) an integrated software productivity improvement program can have an extremely large payoff (a factor of 4 by 1990); (b) improving software productivity involves a long sustained effort; (c) in the very long run, the biggest productivity gains will come from increased use of existing software; (d) software support environment requirements are still too incompletely understood to specify precisely; (e) no single software support system architecture will be optimal for all organizations; (f) a rapid-prototyping capability is essential to the evolutionary development of a software support environment; user-interface standards are essential for preserving the conceptual integrity of an evolving support system; and (g) user acceptance of novel development environments is a gradual process which requires careful nurturing by the sponsoring organization. Also see [Boeh82] and [Bita85].

[Boeh84] Boehm, Barry W. et.al. "A Software Development Environment for Improving Productivity." Computer, Vol. 17, No. 6, June 1984.

Paper presents background and status of TRW's Software Productivity System (SPS). Includes discussion of a 1980 software productivity study. TRW corporate motivation for the study stemmed from: increased demand for software, limited supply of software engineers, rising expectations of software capabilities, and anticipation of reduced costs for computer hardware. The productivity study recommended that TRW initiate a long-range effort to develop a corporate software development environment. In the short-term, the study recommended development of a prototype environment.

The architecture and components of the prototype developed (called SPS-1) included: the work environment (improved office conditions for software engineers); the hardware (a network of VAX's, LSI-11/23's, terminals, and communication equipment); a master project database (composed of a hierarchical file system, a source code control system, and a relational database); general utilities (menu, screen editor, forms package, date/time, report writer); office automation and project support (tool catalog, mail system, text editor/formatter, calendar, forms management interoffice correspondence package); and software development tools (requirements traceability tool, SREM, program design language, Fortran-77 analyzer). TRW's experience in using the prototype showed a definite improvement in productivity. Immediate access to a good set of tools had the highest payoff of all factors studied. Also see [Boeh82] and [Bita85].

[Boeh86] Boehm, Barry W. "A Spiral Model of Software Development and Enhancement." ACM Special Interest Group on Software Engineering Software Engineering Notes, Vol 11, Nr. 4, Aug. 1986, pp. 14-24. Keynote presentation given at International Workshop on the Software Process and Software Environments, Coto de Caza, Trabuco Canyon, California, 27-29 March, 1985.

This is a very important paper describing attempts to further the current state of practice at TRW. The paper describes the software process as an iteration of four phases of activity that could be adapted to fit a variety of approaches and methods. Audience reaction was that this was a "metamodel." Author described how the spiral model was used in developing the TRW Software Productivity System (SPS). Essentially, the spiral model makes four separate "rounds" of the spiral, during each of which a prototype is developed. The first three rounds are: Concept of operation; Software requirements; and Software product design. The fourth round, which begins with an operational prototype, continues through development and acceptance test.

[Broo75] Brooks, Frederick P., Jr. The Mythical Man-Month: Essays on Software Engineering. Reading, Massachusetts: Addison-Wesley Publishing Company. 1975.

Classic in the field. Wise and readable. For example, Brooks is the source of these insights:

a) "Adding manpower to a late software project makes it later" [p. 25]

b) ". . . plan to throw one away; you will, anyhow." [p. 116].

c) "Plan the system for change." [p. 117].

d) "The second-system effect . . . is a tendency to refine techniques whose very existence has been made obsolete by changes in basic system assumptions." [p. 56].

Also see [Fox82].

[Broo84] Brooks, Stephen H. "Macroeconometric Models: Theory and Practice." One of a series of papers in applied business economics commissioned by the National Association of Business Economists. November 1984.

Paper highlights assumptions, construction, and limitations of seven widely-cited econometric models of the total U.S. economy. The seven models are:

a. Commerce Department (Bureau of Economic Analysis) [1000 equations, demand-driven with some monetarist and supply-side approaches];

b. Chase Econometrics [697 equations, demand-driven];

c. CitiSim (Citibank) [224 equations, monetarist];

d. Data Resources, Inc. [1247 equations, demand-driven];

e. Federal Reserve Board ("MPS" Model) [475 equations; demand-driven];

f. Townsend-Greenspan [1450 equations, demand-driven];

g. Wharton Econometric Forecasting Associates [1300 equations, demand-driven].

[BusW84] "Japan's Push to Write 'World-Class' Software." Business Week. February 27, 1984. Pp. 96-98.

Argues that the Japanese are now as dedicated to the computer software market area as they were to consumer electronics and automobiles. Notes that "because of the legendary thoroughness of Japanese workers, 'the finished product here is better, more reliable, and easier to maintain.'"

[Cont86] Conte, S.D., Dunsmore, H.E., and Shen, V.Y. Software Engineering Metrics and Models. Menlo Park, California: Benjamin/Cummings Publishing Company. 1986.

Based on over ten years of work by the Software Metrics Research Group at Purdue University, this book describes software metrics and models. It also suggests how metrics and models can be used by managers, analysts and programmers to help develop more reliable and cost-effective software.

[Curt86] Curtis, Bill, Soloway, Elliot M., Brooks, Ruven E., Black, John B., Ehrlich, Kate, and Ramsey, H. Rudy. "Software Psychology: The Need for an Interdisciplinary Program." Proceedings of the IEEE. August 1986, pp. 1092-1106.

Authors argue that software psychology must identify fundamental characteristics of human behavior involved in interacting with software, and develop means to work with them.

[Dela77] Delaney, William A. "Software Managers Speak Out." Datamation. October, 1977, pp. 77-78.

Article describes typical reactions of software managers of a decade ago, and cites productivity rates typical at the time.

[DoD83] U. S. Department of Defense. "Software Technology for Adaptable, Reliable Systems (STARS) Joint Task Force Report." 15 March 1983. Reprinted in ACM SIGSOFT Software Engineering Notes, Vol. 8, No. 2, April 1983. Also see [IDA84a, IDA84b]

Report began by noting that "the U.S. has lost its lead in many of the mature technologies upon which our industrial base and military power were built." It then described the new STARS program in considerable depth. For an update, see [Lieb86].

[DoD85] U. S. Department of Defense, Defense Financial and Investment Review (DFAIR). "[Report of] Defense Financial and Investment Review [team]," June 1985.

The DFAIR team was chartered to study contract pricing, financing and profit policies to determine if they are resulting in effective and efficient spending of public funds and maintaining the viability of the defense industrial base, as well as to make recommendations for improvements. This is the second study, comparable to the earlier "Profit '76" study.

[Duij1983] Duijn, Jacob. J. van. The Long Wave in Economic Life. London: George Allen & Unwin. 1983.

Book discusses the role of innovation in "long wave" (50-60 year, or Kondratieff) economic cycles. Research on the long wave has traditionally been done in Europe. Author shows that timing of innovations is related to economic growth, particularly to the (7-11 year) capital investment cycle and to the 50-60 year cycle. Author notes that labor-saving innovations can be expected to continue to dominate economic growth in the electronic computer sector during the 1973-1995 Kondratieff "downswing." He notes the long time lag (often more than 20 years) between dates of "invention" and "innovation" (i.e., implementation of an invention).

[Fox82] Fox, Joseph M. Software and Its Development. Englewood Cliffs, Prentice-Hall. 1982.

This is one of two books written by people who have actually managed software development projects (the other, of course, is Brooks's Mythical Man Month). In this book, Fox draws on his experience as manager of IBM's Federal Systems Division. His account differs from most literature, in that he reports the system and software life cycle as it appears from the top, not as he imagines it or wishes it were. His insights are extremely valuable for those who wish to understand the "big picture" of USAF software management and productivity improvement. Also see [Broo75].

[Fren85] Frenkel, Karen A. "Toward Automating the Software-Development Cycle." Communications of the ACM, Vol. 28, No. 6, June 1985, pp. 578-589.

Author surveys expert systems approaches to raising software productivity. She concludes that ". . . the pressure to increase productivity and avoid a shortage of software engineers is a major factor in driving expert-system projects. Researchers are exploring available systems, or are building their own improved versions, to relieve software developers of tedious or time-consuming tasks. . . . Even if expert systems are just another interim technology, it is likely that they and the other approaches together will influence further work that will achieve a significant increase in productivity."

[FSTC86] U.S. General Services Administration, Office of Software Development and Information Technology, Federal Software Testing Center. Software Aids and Tools Survey. Report OIT/FSMC-86/002. November, 1985.

Report lists several hundred software productivity aids and tools that were available in late 1985, primarily for the most popular hardware and programming languages. Some are directly applicable to MCCR software. The products are indexed by software vendor, name, source language, host computer, stage of life cycle, and conversion tool category. The preface cautions that "These products have not been tested or validated by FSMC." (p. iv). Many of the software products have more than 1000 users. Similar (and much more detailed) catalogs are published by commercial firms such as Auerbach and Datapro 70.

[Gilb86] Gilb, Tom. "Estimating Software Attributes: Some Unconventional Points of View." In ACM SIGSOFT Software Engineering Notes, vol. 11, no. 1, Jan 1986, pp. 49-59.

An important paper for program managers. It gives a point of view supplementary to those of Boehm, Halstead, McCabe and Putnam. His emphasis is on how to get control over the results of software projects. He presents ten "principles of estimation." To show the flavor of his principles, here are two.

> "4. Ask me not what the cost will be.
> Design the price you want to see."

> "9. Estimation is a losers' game.
> Action wins your vital aim."

[Goul85] Gould, John D., and Lewis, Clayton. "Designing for usability: key principles and what designers think." Communications of the ACM, Vol 28, Nr. 3., March 1985, pp. 300-311.

Authors describe three principles which they believe must be followed to produce a useful and easy to use computer system. The principles are: early and continual focus on users; empirical measurement of usage; and interactive design whereby the system (simulated, prototype, and real) is modified, tested, modified again, tested again, and the cycle is repeated again and again.

They contrast this approach to other principled design approaches. For example, "get it right the first time," reliance on design guidelines. They present data which show that their design principles are not always intuitive to designers. They identify arguments which designers often offer for not using their principles, and answer them. Finally, they give an example in which their principles were used successfully.

[Hami86] Hamilton, Margaret H. "Zero-defect software: the elusive goal." IEEE Spectrum, March 1986, Vol. 23, Nr. 3, pp. 48-53.

Author argues that zero-defect software is possible in theory, but difficult to achieve. Most common errors are logic and interface, but errors in user intent also occur. Author surveys possible ways to overcome each type of error, and is positive about prospects for success.

[Hans85] Hanson, Stephen Jose, and Rosinski, Richard R. "Programmer Perceptions of Productivity and Programming Tools." Communications of the ACM, Vol. 28, No. 2, February 1985, pp. 180-189.

Paper describes a study using preference scaling methods to determine from the point of view of the programmer, which

programming tools would most favorably affect design and development of software. Sample was not working with MCCR software. Participating programmers were experienced COBOL programmers from a major programming project at Bell Laboratories. A major factor was whether programmers were Bell employees (more familiar with a large range of software tools) or contractors.

Sample size and composition were limited, but conclusions are of interest. Minimal tools are interactive debugger and screen editor. Next phase tools suggested are data dictionary and automatic test generator. Third phase tools add process monitors/meters and source beautifier. The authors suggest a high degree of tool substitutability exists.

[Hara85] Harrison, Richard, and Zvegintzov, Nicholas. "How the software workbench saved Christmas." Datamation, December 15, 1985, pp. 61-65.

Paper describes efforts by the Federal Software Management Support Center to develop Programmer's Workbench Demonstrations. Also refers to vendors, including: Softool Corp., Motorola Four-Phase Systems, and Rand Information Systems Inc.

[Hech82] Hecht, Herb. The Introduction of Software Tools. National Bureau of Standards Special Publication 500-91. September 1982.

This special NBS/ICST publication discusses specific needs for software tools in programming for management information systems and for scientific applications. Steps for the successful introduction of tools are discussed and measures are described to deal with organizational obstacles and difficulties posed by existing computer installations.

[Horo84] Horowitz, Ellis, and Munson, John B. "An Expansive View of Reusable Software." IEEE Transactions on Software Engineering, Vol. SE-10, No. 5, September, 1984. pp. 477-487.

Paper surveys reusability of software, as well as application generators and other potential tools. Authors cite four nontechnical related issues: contractual problems; programmer training; facilitation within the government; and maintenance and modification of systems composed of reusable code elements.

[Houg82a] Houghton, Raymond C., Jr. Software Development Tools. National Bureau of Standards Special Publication 500-88. March, 1982.

Data base of more than 400 software tools, classified according to the tool taxonomy given in FIPS PUB 99. Appendices print the database in different sequences, such as: language written in, and hardware tools are intended for. Data are current as of 1982. Updates to the data base are provided to Rome Air Development Center, but are not classified according the FIPS 99 taxonomy. RADC provides up-to-date printouts.

[Houg82b] __________. "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)." National Bureau of Standards. NBSIR 82-2625. December, 1982.

A review of the Ada Programming Support Environment (APSE), the Army's Ada Language System (ALS), and the Navy's Ada Integrated Environment (AIE), based on the FIPS99 software tool taxonomy. The document includes a comparison of features in the categories of management, static analysis, dynamic analysis, transformation, and input/output. Defines a set of underlying tool primitives that support these features.

[Houg85] __________. and Wallace, Delores R. "Characteristics and Functions of Software Engineering Environments." U.S. Department of Commerce, National Bureau of Standards paper number NBSIR 85-3250.

An important paper. Provides examples of existing software engineering environments now available commercially or in research laboratories.

[Howd82] Howden, William E. "Contemporary Software Development Environments." In Communications of the ACM. Vol. 25, No. 5. May, 1982.

Paper proposes four levels of tool support that could be provided by software engineering environments. For each level, details the type of project, the estimated cost, and the support provided. For example, Environment I has an estimated cost of $35,000 and is for medium-sized projects. Environment IV is estimated to cost $3 million and is for large-scale projects. For each environment level, the author presents tools and techniques for: requirements, design, coding, verification, and management.

[Hunk81] Huenke, H., Editor. Software Engineering Environments. Amsterdam: North-Holland. 1981.

Book, which contains proceedings of a symposium held at Lahnstein, Federal Republic of Germany, in June 1980, is still of immense interest. Papers include [Mats81]. Other papers discuss issues and tools related to software engineering environments, including functional aspects of environments, computer aided design, support for concurrent and distributed system, human factors, description languages, productivity, formal verification, performance, system decomposition and version control. Book concludes with a bibliography by Hausen, Mullerburg and Riddle containing more than 350 citations from 1968 to 1980.

[Ichb84] Ichbiah, J., "Ada: Past, Present, Future." In Communications of the ACM. Vol. 27, Nr. 10, October 1984, pp. 991-997.

The article describes the genesis, conception and current reality of Ada, and is outlined in the form of an interview with the Principal Designer of the Ada language.

[IDA84a] Redwine, Samuel T., Jr., Becker, Louise Giovane, Marmor-Squires, Ann B., Martin, R.J., Nash, Sarah H., and Riddle, William E. Dod Related Software Technology Requirements, Practices, and Prospects for the Future. IDA Paper P-1788. Washington, D.C.: Institute for Defense Analyses. June 1984.

Excellent collection of papers relating to MCCR software.

[IDA84b] DeMillo, Richard A., Marmor-Squires, Ann B., Redwine, Samuel T., Jr., and Riddle, William E. Software Engineering Environments for Mission Critical Applications -- STARS Alternative Programmatic Approaches. IDA Paper P-1789. Washington, D.C.: Institute for Defense Analyses. August 1984.

Collection of papers relating to software engineering environments to be used for MCCR software.

[JLC84] Joint Logistics Commanders' Workshop. "Final Report of the Joint Logistics Commanders' Workshop on Post Deployment Software Support (PDSS) for Mission-Critical Computer Software, Vol. I - Executive Summary." June 1984.

Account of discussions relating to problems of post-deployment support of MCCR software, and potential solutions.

[Jone86] Jones, Capers. Programming Productivity. New York: McGraw-Hill Book Company. 1986.

The latest book on the subject. Author gives many examples.

[Kend77] Kendrick, John W. Understanding Productivity: An Introduction to the Dynamics of Productivity Change. Baltimore: Johns Hopkins University Press. 1977.

Basic description of productivity, from an economist's point of view.

[Laud86] Laudon, Kenneth C. "Data Quality and Due Process in Large Interorganizational Record Systems." Communications of the ACM, Vol. 29, Nr. 1, January 1986, pp. 4-11.

Author assessed several federal- and state-maintained criminal record systems, and found high levels of inaccurate, ambiguous, and incomplete information. He found that a sample of records from the FBI "Ident" system contained only 25.7 percent that were complete, accurate, and unambiguous. 74.3 percent showed some significant quality problems. This is an extreme case, with federal and state organizations both cooperating and competing with one another. Author explains that "Given this uncertain political environment, the FBI has imposed few data-quality controls over participants in its criminal-record systems."

[Ledg82] Ledgrad, M. F., and Singer, A. "Scaling Down Ada." Communications of the ACM. Vol. 25, No. 2, February 1982, pp. 121-5.

Article stresses that, though Ada is an ambitious programming language, its size and complexity may plague its technical success. It presents means of streamlining the language and providing an authorized subset in an effort to scale it down.

[Lieb86] Lieblein, Edward. "The Department of Defense Software Initiative -- A Status Report." Communications of the ACM. Vol. 29, No. 8. August 1986. Pp. 735-744.

Paper describes present status of major Dod software efforts. The "Defense Software Initiative" consists of three major, closely related programs: the Ada standard high-order language program; STARS [Software Technology for Adaptable Reliable Systems]; and the Software Engineering Institute. The massive program includes several demonstration and reusability projects of interest for this project: "Common Ada Missile Packages (CAMP)"; "Ada Based Integrated Control System (ABICS)"; and the "Automation of User Requirements" project.

[Lutt84] Luttwak, Edward N. The Pentagon and the Art of War. New York: Institute for Contemporary Studies/Simon and Schuster. 1984.

Author presents a "devil's advocate" argument, intentionally presenting an adversary view of organizational constraints which [unintentionally, as unforeseen side effects] interfere with and delay development, acquisition, fielding and support of technology in the U.S. Dept. of Defense. Sections relevant to USAF MCCR software are found in pp. 89-91, 166-182, and 218-219.

[Lyon85] Lyons, Michael L. "The DP Psyche." Datamation, August 15, 1985, pp. 103-110.

Article describes an international survey of programmer personality types, based on the Myers-Briggs typology. Also see [Sitt84].

[Mark83] Markus, M. Lynne. "Power, Politics, and MIS Implementation." Communications of the ACM, Vol. 26, No. 6, June 1983, pp. 430-443.

Paper discusses organizational implications of MIS implementation.

[Mats81] Matsumoto, Y., et.al. "SWB System: A Software Factory." In Software Engineering Environments. H. Huenke, Editor. Amsterdam: North-Holland. 1981.

It is difficult to overemphasize the importance of this paper for production of MCCR software. Authors discuss a self-contained large scale "software factory" that produces software for critical applications (nuclear power station controls). **Both productivity and quality of the software products are far superior to present U.S. practice.**

The "factory," operated by Toshiba Corp.'s Heavy Apparatus Engineering Laboratory, consists of three physical buildings, 2000 employees, a methodology, a software environment, intense and systematic efforts to reuse tested code, career paths, and lifetime employment, as well as its own personnel management, finance, and quality control organizations.

The software environment consists of a number of tools and techniques that emphasize the latter of the life cycle (language and library processors, editors, debuggers, etc.). The plans for the environment include addition of tools for the front end (SADT, design languages, etc.).

[McIl68] McIlroy, M.D. "Mass-produced software components." In Software Eng. Concepts and Techniques, 1968 NATO Conf. Software Eng., Buxton, J. M., Naur, P, and Randell, B, eds., 1976, pp. 88-98.

This is the earliest citation on the subject of reusable code.

[Muns81] Munson, John B. "Software Maintainability: A Practical Concern for Life-Cycle Costs." IEEE Computer. November 1981, pp. 103-109.

An excellent, relatively early, discussion of the need to improve post-deployment software support.

[Myer84] Myers, Ware. "Can Software development processes improve--drastically?" IEEE Software, July 1984, pp. 101-102.

A short, philosophical, note that may be of substantial long term value.

[Myer85] __________. "An Assessment of the Competitiveness of the United States Software Industry." In Computer. March, 1985, pp. 81-92.

Also see [Zelk84].

[OTA86b] U. S. Congress, Office of Technology Assessment. Research Funding as an Investment: Can We Measure the Returns? -- A Technical Memorandum. OTA-TM-SET-36, April 1986.

Paper discusses difficulties of measurement in this field.

[Pind81] Pindyck, Robert S., and Rubinfeld, Daniel L. Econometric Models and Economic Forecasts, 2nd Ed. New York: McGraw-Hill Book Company. 1981.

A thorough and rigorous treatment of the field. Authors both received their Ph.D. degrees as M.I.T., where Pindyck is Professor of Applied Economics at the Sloan School of Management. Rubinfeld is Assoc. Professor of Economics and Law at the University of Michigan.

[Rade85] Rader, Jock A. "Experiences Building and Using a Software Engineering Environment for Avionics Software." IEEE paper CH21352/85/0/0358$1.00.

Paper discusses experience in building a using an environment in building MCCR software for radar applications at Hughes Aircraft Company. Author describes preliminary experience. In his view, reusable software is not as practical as has been expected.

[Ridd86] Riddle, William E., and Williams, Lloyd G. "Software Environments Workshop Report." ACM SIGSOFT Software Engineering Notes, Vol. 11, Nr. 1, Jan. 1986, pp. 73-102.

Paper describes a workshop, co-sponsored by the National Science Foundation and the Rocky Mountain Institute of Software Engineering.

[Samps84] Sampson, Charles H, Fritz, Robert E., Gillen, Michael J., *Olson*, *Alan* R., *Sans*, Conway C., and Fisher, Gerald A., Jr. "Developing an Ada to CMS-2 Translator." Proceedings of IEEE Computer Society 1984 Conference on Ada Applications and Environments, October 15-18, 1984, pp. 83-88. IEEE Computer Society Press. IEEE Catalog No. 84CH2083-4.

Discusses an approach to the problem of Dod-wide standard software support environments. CMS-2 is a primary language used by the U.S. Navy, which has a large inventory of MCCR software written in the language.

[Silv85] __________. "Software Cost and Productivity Improvements: An Analogical View." Computer. May 1985, pp. 86-96.

Important article for those working with MCCR software. Author studied ways in which experienced software people refer to previous projects ("analogies") in designing new software. Author demonstrates that, by making small adjustments in information flows and job designs, software managers can reuse existing software more effectively. Among other recommendations, author suggests including this activity in procedures governing software life cycle, to permit designers to include this process explicitly.

[Sitt84] Sitton, Sarah, and Chmelir, Gerard. "The Intuitive Computer Programmer." Datamation. October 15, 1984, pp. 137-140.

Certain types of personality seem to be drawn to the occupation of computer programmer. This paper reports on a study in which typical characteristics were found and inferences drawn. Using the Myers-Briggs Type Indicator, authors worked with a sample of 27 volunteers in Texas. The most common personality type among [the sample of] ADP people is ENTP [extrovert, intuitive, thinking, perceiving]. Only about five percent of the general population is of this type.

The occupation best suited to this personality type is that of inventor. The type is ". . . good at analysis, especially functional analysis, with a tolerance for and enjoyment of the complex . . . always looking for new projects, new activities, new procedures. . . . They ignore the standard, the traditional, and the authoritative." Authors comment that "this personality type places a high value on innovation and may rely on ingenuity to carry the day, sometimes neglecting necessary preparation. As soon as a problem is solved they may lose interest--unless they are very self-disciplined--and simply walk away from the project. Also see [Lyon85].

[Stan84] Standish, Thomas A. "An Essay on Software Reuse." IEEE Transactions on Software Engineering, Vol. SE-10, No. 5, Sept., 1984, pp. 494-497.

Paper explores software reuse. It discusses briefly some economic incentives for developing effective software reuse technology and notes that different kinds of software reuse, such as direct use without modification and reuse of abstract modules after refinement, have different technological implications. It then sketches some problem areas to be addressed if we are to achieve the goal of devising practical software reuse systems. These include information retrieval problems and finding effective methods to aid us in understanding how programs work. Author presents a philosophical epilogue which stresses the importance of having realistic expectations about the benefits of software reuse.

[Stue84] Stuebing, H.G. "A Software Engineering Environment (SEE) for Weapon System Software." In IEEE Transactions on Software Engineering, Vol. SE-10, No. 4, July 1984.

Paper presents a large scale environment called FASP that is hosted on multiple, large scale commercial computers. FASP primarily supports the latter stages of software development, but the author also discusses extension to the requirements and design phases. The author attributes a two-fold increase in productivity (lines per month) to FASP, and in increase in software quality due to the tools, standards, and testing associated with the FASP environment. FASP includes an underlying database made up of libraries: Source library, object library, test library, interface library, production data library, and documentation library. The system is command driven; the commands consist of lower level system commands (command procedures). Testing is supported by the Automated Testing Analyzer (ATA) and there is support for multiple languages and target computers.

[Taj84] Tajima, Denji, and Matsubara, Tomoo. "Inside the Japanese Software Industry." Computer. March 1984, pp. 34-43.

Paper describes the Japanese software industry, focusing on Hitachi Engineering [HSK]. It is written from the viewpoint of the Japanese, showing how that nation's philosophies, ways of living, home and business environments affect the ways they "manufacture" their software.

[USGA86] U. S. General Accounting Office. "Government Equipment: Defense Should Further Reduce the Amount It Furnishes to Contractors." GAO/NSIAD-86-109, June 19, 1986.

GAO believes that the military services and defense agencies should be allowed to provide general purpose equipment to contractors only under highly unusual circumstances which are clearly defined, adequately controlled, and properly justified. Specific guidelines should be developed for program managers and contracting officials to use in determining when and under what conditions the government can provide general purpose equipment to service contractors.

[Wall84] Wallich, Paul. "A review of engineering workstations." IEEE Spectrum. Oct. 1984, pp. 48-53.

Review of engineering workstations two years ago.

[Werl83] Werling, P.R. Alternative Models of Organizational Reality: The Case of Public Law 89-306 [The Brooks Act]. D.P.A. Dissertation submitted to the University of Southern California School of Public Administration. 1983.

Paper addressed two major issues. In the first, economic aspects of computing in the Federal government, author shows that computing cost/performance improved by more than 30 percent annually in the years 1958-1980. In these years, a new "generation" of computing technology emerged every five years.

In the second, the author questioned why the Federal government fell farther and farther behind the state of the computing art after enactment of the Brooks Act in 1966. By 1980 Federal computers averaged five years older than those in the private sector, and were much less productive economically. Difficulties were traced to fundamental discrepancies among three models of organizational reality that describe behavior of separate groups of public servants: a) operating agencies [such as USAF] act in accordance with the "organizational process" model, taught in the school of hard knocks; b) the General Accounting Office and regulatory agencies exhibit expectations and behavior that correspond to the "classical management" model (taught in business schools); and c) authors of legislation and implementing procedures function as though guided by the "adversary proceedings" model (taught in law schools).

Tests of seven substantive hypotheses showed that the "classical management" model is useful for the first estimate of results, while the "organizational process" and "adversary proceedings" models provide valuable insights for anticipating dysfunctions in implementation.

[Werl85] __________, Houghton, R. C., Jr., and Chande, A. M. "Use of a Software Development and Support Environment as Government-Furnished Equipment (GFE)." Report of research conducted for U.S.A.F. Business Research Management Center, AFBRMC/RDCB, Wright-Patterson AFB, OH. June 28, 1985. The research was supported under contract #F 33615-84-C-5114.

Paper describes: (1) what an integrated automated software development/support environment should consist of; (2) what software tools and methods were available [in 1985], and what needed to be developed for the integrated automated environment to be feasible; and (3) pros and cons of developing a standard environment to be provided as Government-Furnished property. Conclusions of the research were:

(a) USAF <u>can</u> now build an integrated, automated software development/support environment with [1985] technology;

(b) More than 400 software tools were available [usually written in FORTRAN, rather than in Ada], with at least one tool to support each function needed during the life cycle of software for mission-critical systems;

(c) Improvements in productivity, ranging from ten to twenty percent annually, can probably be sustained by USAF contractors who use such an environment;

(d) Use of a standard environment would improve the post-deployment maintenance and enhancement of mission-critical software;

(e) A standard environment must be designed to accommodate significant future changes in software modules, user interface, and methodology. That is, the functional capabilities built into an environment must be designed for frequent change throughout the entire life cycle of the environment; and

(f) Because of the complexity of the issues, a detailed econometric model is needed to establish the cost-effectiveness of such a GFE program.

[Werl86] __________. "Data Collection System for Estimating Software Development Cost." Report of research conducted for U.S.A.F. Business Research Management Center, AFBRMC/RDCB, Wright-Patterson AFB, OH. September 1986. The research was supported under contract number F 33615-85-C-5123.

Paper describes research and development of a data collection system to help USAF Contract Management Division [AFCMD] resolve difficulties encountered in obtaining estimates of costs and schedules for software development. AFCMD technical evaluators need help in: (a) obtaining accurate estimates for software before development by contractors; and (b) validating estimated costs and schedules submitted by contractors. While contractors reported productivity increases of more than ten percent per year, USAF contracts did not reflect those gains.

Paper describes how contractor developed: (a) a PC-based data collection program that permits AFPRO specialists to encode essential data on their software projects; and (b) a prototype computer program for AFCMD specialists to use in evaluating contractors' proposals for new software projects. The prototype computer program, based on the widely used "COCOMO" cost estimating model, separates cost drivers into two broad categories. The first category of costs, primarily under the control of the government, consists of project characteristics that are Requirements-Driven. The second category, primarily controllable by contractors, involves such cost drivers as level of experience, use of software tools, and turnaround time of the development computer.

Project staff visited with AFCMD specialists and with contractors' cost estimators at six Air Force Program Review Offices [AFPROs], to ensure understanding of the wide variety of AFPRO functions and responsibilities. Contractor designed a data collection system that provides input data required by four popular models and is compatible with a VAX-based system in use at Hanscom AFS, MA.

[Wolf84] Wolf, Alexander L., Clarke, Lori A., and Wileden, Jack C. "An Ada Environment for Programming-in-the-Large." Proceedings of IEEE Computer Society 1984 Conference on Ada Applications and Environments, October 15-18, 1984, pp. 52-62. *IEEE Computer Society Press*. IEEE Catalog No. 84CH2083-4.

[Zelk84] Zelkowitz, M.V., et.al. "Software Engineering Practices in the U.S. and Japan." Computer. June 1984, pp. 57-65.

Paper gives results of an in-depth survey of 30 companies. It shows that software development/support practices are ten years behind software engineering research. Authors suggest that tools now available can narrow this gap.

[Zveg84a] Zvegintzov, Nicholas. "Immortal Software." Datamation, June 15, 1984. Pp. 170-180.

Paper argues that systems grow old, but rarely die. Author recommends "structured retrofit" as a technique for keeping systems vigorous.

[Zveg84b] __________. "Front-End Programming Environments." Datamation. Aug. 15, 1984, pp. 80-88.

Paper describes several popular programming environments, then marketed for the commercial software industry.

APPENDIX A

CHARACTERISTICS OF ENVIRONMENTS

The following pages summarize the survey conducted by Raymond C. Houghton, Jr. The entries contain information required for the econometric model, in some cases coded consistent with FIPS 99.

ENVIRONMENT SUMMARY

Environment	Functionality	Size	Cost/ 1	10	100 Sites
MAPSE					
1 Average MAPSE	49%	450K	$ 18,900K (development cost)		
2 ALS/VMS	38%	500K	$ 22,000K (development cost)		
3 VADS/VMS	36%	700K	$ 20,000	$ 8,000	$ 2,000
4 R-1000 (Rational)	36%+	1000K	$600,000		
5 Harris (HAPSE)	36%		$ 50,000		
6 FASP	44%	600K+		GFE	
APSE					
7 Average APSE	84%	830K	$ 80,000K (development cost)		
8 STARS-SEE	82%	150MEG*	$150,000K (development cost)		
9 SPS/Unix	80%				
10 SOFTOOL/VMS	53%	1.5MEG*	$ 60,000	$ 24,000	
11 ARGUS I/Unix	53%	70K	$ 50,000		
Baseline OS					
Unix	29%	13K (Kernel) 16MEG* (Support)			
VAX/VMS	22%	250K			
CDC Cyber	20%	700K			

* estimated

Environment Index Number	1
Environment Name	Average MAPSE
Vender/Developer	STONEMAN
Environment Type	MAPSE
Number of Languages Supported	1
Functional Support (Functions/45)	22/45 = 49%
Underlying Operating System	real-time, time sharing, or virtual memory operating system
Size (KDSI)	450
Development Cost	$18,900,000
Purchase Cost (1 Site)	
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	
Version Life (Months)	
Contact	
References	Barry W. Boehm, "Software Engineering Economics", Prentice-Hall, 1981

FUNCTIONALITY

Index No. 1

TRANSFORMATION

[✓ T1]	Editing
T2	Formatting
[✓ T3]	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
[✓ D3]	Coverage Analysis
[✓ D4]	Resource Utilization
D5	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
[✓ D8B]	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
D10	Regression Testing

STATIC ANALYSIS

S1	Auditing
S2	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
[✓ S7]	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
S10	Scanning
[✓ S11]	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

G1	Configuration Management
	(G2 Information Management)
G2A	Data Dictionary Management
G2B	Documentation Management
[✓ G2C]	File Management
[✓ G2D]	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	2
Environment Name	ALS (Ada Language System)
Vender/Developer	Softech, Inc., Waltham, MA
Environment Type	MAPSE
Number of Languages Supported	1
Functional Support (Functions/45)	17/45 = 38%
Underlying Operating System	VAX/VMS
Size (KDSI)	500
Development Cost	$22,000,000
Purchase Cost (1 Site)	$1200 (Reproduction Cost)
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	3
Version Life (Months)	6
Contact	Beverly Vidler, 617-890-6900
References	Softech Literature

FUNCTIONALITY

Index No. 2

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
T3	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
D3	Coverage Analysis
D4	Resource Utilization
D5	Simulation
D6	Symbolic Execution
D7	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
D10	Regression Testing

STATIC ANALYSIS

S1	Auditing
[✓ S2]	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
S10	Scanning
S11	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
G2A	Data Dictionary Management
G2B	Documentation Management
[✓ G2C]	File Management
G2D	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	3
Environment Name	VADS (Verdix Ada Development System)
Vender/Developer	Verdix Corp., Aloha, OR
Environment Type	MAPSE
Number of Languages Supported	1
Functional Support (Functions/45)	16/45 = 36%
Underlying Operating System	VAX/VMS
Size (KDSI)	700 (Includes Generation Code)
Development Cost	
Purchase Cost (1 Site)	$20,000
Purchase Cost (100 Sites)	$8,000
Purchase Cost (1000 Sites)	$2,000
Number of Versions	
Version Life (Months)	
Contact	David Schwartzman, 703-378-7600
References	Verdix Literature

FUNCTIONALITY

Index No. 3

TRANSFORMATION

[✓ T1]	Editing
T2	Formatting
T3	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
T6D	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
D3	Coverage Analysis
D4	Resource Utilization
D5	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
D9	Tuning
D10	Regression Testing

STATIC ANALYSIS

S1	Auditing
S2	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
S10	Scanning
S11	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
G2D	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	4
Environment Name	R-1000 Development System
Vender/Developer	Rational Inc., Palo Alto, CA
Environment Type	MAPSE
Number of Languages Supported	1
Functional Support (Functions/45)	16/45 = 36% (plus an Ada-oriented Interface)
Underlying Operating System	None
Size (KDSI)	1,000 (1 Meg)
Development Cost	
Purchase Cost (1 Site)	$600,000
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	
Version Life (Months)	
Contact	Jim Hake, 415-940-4761
References	Rational Literature

FUNCTIONALITY

Index No. 4

TRANSFORMATION

[✓ T1]	Editing
T2	Formatting
T3	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
T6D	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
D3	Coverage Analysis
D4	Resource Utilization
D5	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
D9	Tuning
D10	Regression Testing

STATIC ANALYSIS

S1	Auditing
S2	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
S10	Scanning
S11	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
G2D	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	5
Environment Name	HAPSE (Harris Ada Programming Support Environment)
Vender/Developer	Harris, Inc., Melburn, FL
Environment Type	MAPSE
Number of Languages Supported	1
Functional Support (Functions/45)	16/45 = 36%
Underlying Operating Sytem	H-800 or H-1200 (Harris Systems)
Size (KDSI)	
Development Cost	
Purchase Cost (1 Site)	$50,000
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	
Version Life (Months)	
Contact	Bill Marlow, 305-977-5502
References	Harris Literature

FUNCTIONALITY Index No. 5

TRANSFORMATION

[✓ T1]	Editing
T2	Formatting
T3	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
T6D	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
D3	Coverage Analysis
D4	Resource Utilization
D5	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
D9	Tuning
D10	Regression Testing

STATIC ANALYSIS

S1	Auditing
S2	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
S10	Scanning
S11	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
G2D	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	6
Environment Name	FASP (Facility for Automated Software Production)
Vender/Developer	Navel Air Development Center (NADC)
Environment Type	MAPSE
Number of Languages Supported	3+
Functional Support (Functions/45)	20/45 = 44%
Underlying Operating System	CDC Cyber
Size (KDSI)	125K (Envelope) 125K (for each Language)
Development Cost	
Purchase Cost (1 Site)	GFE
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	
Version Life (Months)	
Contact	Tony Williamson, 215-441-3145
References	H. G. Steubing, "A Modern Facility for Software Production and Maintenance", Proceedings of COMPSAC, October 1981

FUNCTIONALITY Index No. 6

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
[✓ T3]	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
[✓ D3]	Coverage Analysis
D4	Resource Utilization
D5	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
D10	Regression Testing

STATIC ANALYSIS

S1	Auditing
[✓ S2]	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
S9	Interface Analysis
S10	Scanning
S11	Statistical Analysis
S12	Structure Checking
S13	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
[✓ G2D]	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	7
Environment Name	Average APSE
Vender/Developer	STONEMAN
Environment Type	APSE
Number of Languages Supported	1
Functional Support (Functions/45)	38/45 = 84%
Underlying Operating System	real-time, time sharing, or virtual memory operating system
Size (KDSI)	830K
Development Cost	$80,000,000
Purchase Cost (1 Site)	
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	
Version Life (Months)	
Contact	
References	Barry W. Boehm, "Software Engineering Economics", Prentice-Hall, 1981

FUNCTIONALITY Index No. 7

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
[✓ T3]	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
[✓ T6C]	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
[✓ D2]	Constraint Evaluation
[✓ D3]	Coverage Analysis
[✓ D4]	Resource Utilization
D5	Simulation
[✓ D6]	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
[✓ D8B]	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
[✓ D10]	Regression Testing

STATIC ANALYSIS

S1	Auditing
[✓ S2]	Comparison
S3	Complexity Measurement
[✓ S4]	Completeness Checking
[✓ S5]	Consistency Checking
[✓ S6]	Cross Reference
[✓ S7]	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
[✓ S10]	Scanning
[✓ S11]	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
[✓ G2D]	Test Data Management
	(G3 Project Management)
[✓ G3A]	Cost Estimation
[✓ G3B]	Resource Estimation
[✓ G3C]	Scheduling
[✓ G3D]	Tracking

Environment Index Number	8
Environment Name	STARS SEE (STARS Softvare Engineering Environment)
Vender/Developer	STARS Systems Group
Environment Type	APSE
Number of Languages Supported	1
Functional Support (Functions/45)	37/45 = 82%
Underlying Operating System	none
Size (KDSI)	150,000K (Estimated)
Development Cost	$150,000,000
Purchase Cost (1 Site)	
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	0
Version Life (Months)	
Contact	Hank Steubing, 213-441-2314
References	STARS SEE Specs

FUNCTIONALITY Index No. 8

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
[✓ T3]	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
[✓ T6C]	Conversion
[✓ T6D]	Macro Expansion
[✓ T7]	Synthesis

DYNAMIC ANALYSIS

[✓ D1]	Assertion Checking
D2	Constraint Evaluation
[✓ D3]	Coverage Analysis
[✓ D4]	Resource Utilization
[✓ D5]	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
[✓ D10]	Regression Testing

STATIC ANALYSIS

[✓ S1]	Auditing
[✓ S2]	Comparison
S3	Complexity Measurement
[✓ S4]	Completeness Checking
[✓ S5]	Consistency Checking
[✓ S6]	Cross Reference
[✓ S7]	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
[✓ S10]	Scanning
S11	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
[✓ G2D]	Test Data Management
	(G3 Project Management)
[✓ G3A]	Cost Estimation
G3B	Resource Estimation
[✓ G3C]	Scheduling
[✓ G3D]	Tracking

Environment Index Number	9
Environment Name	SPS (Software Productivity System)
Vender/Developer	TRW, Inc.
Environment Type	APSE
Number of Languages Supported	4+
Functional Support (Functions/45)	36/45 = 80%
Size (KDSI)	
Development Cost	
Purchase Cost (1 Site)	
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	
Version Life (Months)	
Contact	Inad Bitar, 213-535-4321
References	Barry W. Boehm, et. al., "A Software Development Environment for Improving Productivity", Computer, June 1984

FUNCTIONALITY

Index No. 9

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
[✓ T3]	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

[✓ D1]	Assertion Checking
D2	Constraint Evaluation
[✓ D3]	Coverage Analysis
D4	Resource Utilization
[✓ D5]	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
[✓ D8B]	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
[✓ D10]	Regression Testing

STATIC ANALYSIS

[✓ S1]	Auditing
[✓ S2]	Comparison
[✓ S3]	Complexity Measurement
[✓ S4]	Completeness Checking
[✓ S5]	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
[✓ S10]	Scanning
[✓ S11]	Statistical Analysis
[✓ S12]	Structure Checking
[✓ S13]	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
[✓ G2D]	Test Data Management
	(G3 Project Management)
[✓ G3A]	Cost Estimation
G3B	Resource Estimation
[✓ G3C]	Scheduling
[✓ G3D]	Tracking

Environment Index Number	10
Environment Name	SOFTOOL
Vender/Developer	SOFTOOL Corp., Goleta, CA
Environment Type	APSE
Number of Languages Supported	4
Functional Support (Functions/45)	24/45 = 53%
Underlying Operating System	main-frame or super-mini scale operating system (VAX/VMS)
Size (KDSI)	1,500K (estimated from required disk storage)
Development Cost	
Purchase Cost (1 Site)	$60,000
Purchase Cost (100 Sites)	$24,000
Purchase Cost (1000 Sites)	
Number of Versions	5
Version Life (Months)	6 months
Contact	Leon Presser, 805-683-5777
References	SOFTOOL Literature

FUNCTIONALITY

Index No. 10

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
[✓ T3]	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
[✓ T6C]	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
[✓ D3]	Coverage Analysis
[✓ D4]	Resource Utilization
D5	Simulation
D6	Symbolic Execution
D7	Timing
	(D8 Tracing)
D8A	Breakpoint Control
D8B	Data Flow Tracing
[✓ D8C]	Path Flow Tracing
[✓ D9]	Tuning
[✓ D10]	Regression Testing

STATIC ANALYSIS

[✓ S1]	Auditing
[✓ S2]	Comparison
S3	Complexity Measurement
S4	Completeness Checking
S5	Consistency Checking
[✓ S6]	Cross Reference
[✓ S7]	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
[✓ S10]	Scanning
[✓ S11]	Statistical Analysis
[✓ S12]	Structure Checking
S13	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
G2A	Data Dictionary Management
G2B	Documentation Management
[✓ G2C]	File Management
G2D	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
G3C	Scheduling
G3D	Tracking

Environment Index Number	11
Environment Name	ARGUS I
Vender/Developer	Boeing Computer Systems
Environment Type	APSE
Number of Languages Supported	4
Functional Support (Functions/45)	24/45 = 53%
Underlying Operating System	Unix
Size (KDSI)	70K
Development Cost	
Purchase Cost (1 Site)	$50,000
Purchase Cost (100 Sites)	
Purchase Cost (1000 Sites)	
Number of Versions	2
Version Life (Months)	6-9 months
Contact	
References	Leon G. Stucki, "What about CAD/CAM for Software? The ARGUS Concept", Proceedings of SoftFair, July 1983.

FUNCTIONALITY

Index No. 11

TRANSFORMATION

[✓ T1]	Editing
[✓ T2]	Formatting
T3	Instrumentation
[✓ T4]	Optimization
	(T6 Translation)
[✓ T6A]	Assembling
[✓ T6B]	Compilation
T6C	Conversion
[✓ T6D]	Macro Expansion
T7	Synthesis

DYNAMIC ANALYSIS

D1	Assertion Checking
D2	Constraint Evaluation
D3	Coverage Analysis
D4	Resource Utilization
D5	Simulation
D6	Symbolic Execution
[✓ D7]	Timing
	(D8 Tracing)
[✓ D8A]	Breakpoint Control
D8B	Data Flow Tracing
D8C	Path Flow Tracing
[✓ D9]	Tuning
D10	Regression Testing

STATIC ANALYSIS

[✓ S1]	Auditing
[✓ S2]	Comparison
S3	Complexity Measurement
[✓ S4]	Completeness Checking
[✓ S5]	Consistency Checking
[✓ S6]	Cross Reference
S7	Data Flow Analysis
[✓ S8]	Error Checking
[✓ S9]	Interface Analysis
[✓ S10]	Scanning
S11	Statistical Analysis
[✓ S12]	Structure Checking
S13	Type Analysis
S14	Units Analysis
S15	I/O Specification Analysis

MANAGEMENT

[✓ G1]	Configuration Management
	(G2 Information Management)
[✓ G2A]	Data Dictionary Management
[✓ G2B]	Documentation Management
[✓ G2C]	File Management
G2D	Test Data Management
	(G3 Project Management)
G3A	Cost Estimation
G3B	Resource Estimation
[✓ G3C]	Scheduling
[✓ G3D]	Tracking

OLD EQUATIONS

APPENDIX B

SUMMARY DESCRIPTION OF ECONOMETRIC MODEL

This appendix summarizes the equations used by the Automated Econometric Model*. A more complete explanation is given in the User's Guide. The equations determine costs, benefits, and net benefits involved with implementation of a selected environment (i) using a selected control strategy (j). The environment-control strategy pair is referred to as a scenario (i,j). The model tracks each scenario by year (n) for the period 1986-1995. Thus equations may contain the three subscripts (i,j,n).

The model consists of cost equations, and benefit equations. Net benefits of implementing the scenario (i,j,n) represent the sum of benefits less the sum of costs over the years in the period modeled.

COST EQUATIONS

Eqn 1: **COST(i,j,n) = RC(i,j,n) + OC(i,j,n) + DC(i,j,n)**, where:

COST(i,j,n)	is the cost for implementing scenario (i,j) for year (n);
RC(i,j,n)	is the initial R&D (development) cost of a scenario (i,j) for year (n);
OC(i,j,n)	is the operational cost associated with scenario (i,j) for year (n); and
DC(i,j,n)	is the "disposal" cost to USAF associated with implementing scenario (i,j) for year (n).

Eqn 2: **RC(i,j,n) = ENVRC(i,n) * SWEIGHT(j)**, where:

ENVRC(i,n)	is the initial R&D (development) cost of the selected environment (i) for year (n);
SWEIGHT(j)	is a weighting factor assigned to the selected control strategy (j).

Eqn 3: **OC(i,j,n) = FOC(i,j,n) + VOC(i,j,n)**, where:

FOC(i,j,n)	is the fixed operating cost for a particular scenario (i,j) for a given year (n);
VOC(i,j,n)	is the variable operating cost for a particular scenario (i,j) for given year (n).

* Adapted from Automated Econometric Model User's Guide, June 20, 1986, pp. 3-1 to 3-7. The User's Guide presents a full description of the equations and their implementation using LOTUS 1-2-3.

Eqn 4: **FOC(i,j,n) = FOCENV(i,n) * GOVSIT(j)**, where:

FOCENV(i,n)	is the fixed operating cost associated with selected environment (i) for year (n);
GOVSIT(j)	is the number of government sites operating under the selected control strategy (j).

Eqn 5: **FOCENV(i,n) = OVRHD(i,n) + CONFIG(i,n) + QA(i,n) + DOCSUP(i,n) + TEST(i,n) + PRGSUP(i,n) + TRAIN(i,n)**, where:

OVRHD(i,n)	is indirect cost not directly traceable to the selected environment (i) in given year (n);
CONFIG(i,n)	is cost of configuration management for the selected environment (i) in year (n);
QA(i,n)	is cost of quality assurance for the selected environment (i) for year (n);
DOCSUP(i,n)	is cost of supporting documentation for environment (i) in year (n);
TEST(i,n)	is cost of test support for environment (i) in year (n);
PRGSUP(i,n)	is cost of programming support for environment (i) in year (n);
TRAIN(i,n)	is training cost associated with environment (i) in year (n)

Eqn 6: **VOC(i,j,n) = VOCENV(i,n) * CONSIT(j,n)**, where:

VOCENV(i,n)	is fixed industry operating cost associated with environment (i) for year (n);
CONSIT(j,n)	is the number of contractor sites operating under control strategy) for year (n).

Eqn 7: **CONSIT(j,n) = $L/(1 + a^{-b*n})$, for j = 1,3,5,7, or 9**

= 0 for j = 2,4,6, or 8

where: L	is the upper limit on number of contractor sites;
a	is "Pearl curve" constant [set at about 1000]; and
b	is "Pearl curve" constant [set at about 0.7].

Eqn 8: **DC(i,j,n)** = **BUP(i,n) + ICHG(i,n)**, where:

BUP(i,n) is cost of block upgrades (major changes, new software versions) for environment (i) in year (n).

ICHG(i,n) is cost of incremental changes (minor software changes) for environment (i) in year (n).

Eqn 9: **BUP (i,n)** = **W(n) * RC(i,j,n)**, where:

W(n) is the percentage plowed back into continuing development (R&D) of environment (i) for year (n).

Eqn 10: **ICHG(i,n)** = **P(n) * RC(i,j,n)**, where:

P(n) is the percentage of support cost [RC(i,j,n)] expended for continuing development (i.e., incremental improvements) in environments during year (n).

COST CONSTRAINTS

If the following cost constraints are violated, a warning message is displayed:

Eqn 11: $RC(i,j,n) / RCAP(n) < 1$
Eqn 12: $OC(i,j,n) / OCAP(n) < 1$
Eqn 13: $DC(i,j,n) / DCAP(n) < 1$

where RCAP(n), OCAP(n), and DCAP(n) are the government funds limitations for year (n).

The warning display includes a tabulation of the computed costs and the funds available. It then advises the user that he should assure availability of adequate funds.

BENEFIT EQUATIONS

Eqn 14: **BENFIT(i,j,n)** = **NPC(i,j,n)** + **NAE(i,j,n)** + **NSR(i,j,n)** + **NEE(i,j,n)** + **NTR(i,j,n)**, where:

BENFIT(i,j,n)	is the dollar benefit of implementing a particular scenario (i,j) for year (n);
NPC(i,j,n)	is the dollar benefit of nonproliferation [of different software support environments] for year (n);
NAE(i,j,n)	is the dollar benefit from use of advanced environment (i) for year (n);
NSR(i,j,n)	is the dollar benefit from reuse of existing software, by implementing scenario (i,j) for year (n);
NEE(i,j,n)	is the dollar benefit from users' increased facility [stemming from experience using an advanced environment] in implementing scenario (i,j) for year (n); and
NTR(i,j,n)	is dollar benefit from improved personnel performance associated with faster computer response [turn-around] time of environment (i), using control strategy (j) for year (n).

Eqn 15: **NPC(i,j,n)** = **[FOC(i,j,n-1) + FOCENV(i,n-1) * (ASPS -1)] - [FOC(i,j,n) + FOCENV(i,n) * (ASPS -1)* WSC(j)]**

where:

ASPS	is the average numnber of system environments per government site; and
WSC(j)	is the fractional saving due to a common environment being available throughout a given government site.

Eqn 16:	**NAE(i,j,n)**	= [(1/PLOC(n) - (1/PLOC(n) * COCAE))] * [ACPY(n) * (DIPY(j,n) * CONSIT(j,n)/L + SIPY(j,n)]

where:

PLOC(n)	is the average lines of code delivered per programmer for year (n);
COCOAE	is a multiplier derived from "COCOMO" multipliers TOOL (for degree to which software tools are used) and MODP (for degree to which "modern programming practices" are used);
ACPY(n)	is average (total billed) cost per programmer in year (n), [about $100,000 in FY 1986];
DIPY(j,n)	is the number of delivered source instructions (DSI) developed, operating under control strategy (j) in year (n);
SIPY(j,n)	is the number of delivered source instructions (DSI) supported, operating under control strategy (j) in year (n).

Eqn 17:	**COCOAE**	= [MODPTO * TOOLTO] / [MODP(n) * TOOL(n)], where
	MODPTO	is the COCOMO multiplier MODP (extent to which modern programming pracices are used) before implementing scenario (i,j);
	TOOLTO	is the COCOMO multiplier TOOL (extent to which software tools are used) before implementing scenario (i,j);
	MODP(n)	is the COCOMO multiplier MODP for year (n);
	TOOL(n)	is the COCOMO multiplier TOOL for year (n).

Eqn 18:	**NSR(i,j,n)**	= [(1/PLOC(n) - (1/PLOC(n) * COCSR))] * [ACPY(n) * (DIPY(j,n)) * CONSIT(j,n)/L + SIPY(j,n)];

where:

COCOSR	is a multipler, derived in eqn. 19, for reuse of existing software.

Eqn 19: COCOSR = LIBRTO / LIBR(n), where

LIBRTO	is a multiplier, derived from the COCOMO literature, for "LIBR" (extent to which existing code is reused) before implementing scenario (i,j);
LIBR(n)	is a multiplier representing extent to which existing code is reused in modelling scenario (i,j) in year (n).

Eqn 20: **NEE(i,j,n) = [(1/PLOC(n)) - (1/(PLOC(n) * COCOEE))] * [ACPY(n) * (DIPY(j,n) * CONSIT(j,n)/L + SIPY(j,n)) - COTY(n)]**

where:

COCOEE	is a multipler, derived in eqn. 21, for benefit due to experience using scenario (i,j);
COTY(n)	is the cost of training for scenario (i,j) for year (n).

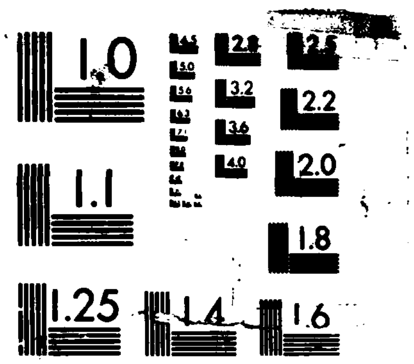
Eqn 21: **COCOEE = [(LEXTO * SENTO) / (LEX(n) * SEN(n))], where**

LEXTO	**is the COCOMO multiplier LEXP (experience with the programming language)** ***before*** **implementing scenario (i,j);**
SENTO	**is the COCOMO-like multiplier for experience with an advanced environment before implementing scenario (i,j);**
LEX(n)	**is the COCOMO multiplier LEXP (experience with the programming language) for year (n) of implement[illegible] scenario (i,j);**
SEN(n)	**is the COCOMO-like multiplier for experience wi[illegible] an advanced environment f[illegible] yea[illegible] implementing scenari[illegible]**

AD-A181 883 COST EFFECTIVENESS TRADE-OFFS IN SOFTWARE SUPPORT ENVIRONMENT STANDARDIZATION(U) TECHNION INTERNATIONAL INC WILMINGTON DE R WERLING 30 SEP 86 BRMC-85-5165-1 2/2

UNCLASSIFIED F33615-85-C-5165 F/G 12/5 NL

END
8-87
DTIC



MICROCOPY RESOLUTION TEST CHART

Eqn 22: COTY(n) = [((CONSIT(n)-CONSIT(n-1)*NIP) / (DIPY(j,n)*DUR)]*ACPY(n)/6+
[CONSIT(n-1)*ACPY(n-1)/6*NIP / (DIPY(j,n)*DUR)] * 0.12

where:

NIP	is the average number of deliverable source instructions (DSI) per program; and
DUR	is the average duration of the program.

NOTE: This equation assumes: (a) programmer turnover rate of 12 percent per year; and (b) training requires two months (i.e., 1/6 of a year).

Eqn 23: **NTR(i,j,n) = [(1/PLOC(n))-(1/PLOC(n)*COCOTR))] * ACPY(n)
*(DIPY(j,n)*CONSIT(j,n)/L + SIPY(j,n)],** where:

COCOTR	is a multipler, derived in eqn. 24, for benefit due to improved computer response (turnaround) time.

Eqn 24: COCOTR = **TURNTO / TURN(n)**, where

TURNTO	is the COCOMO multiplier, TURN, for computer response time, before implementation of scenario (i,j);
TURN(n)	is the COCOMO multiplier, TURN, for computer response time for year (n).

ADDITIONAL EQUATIONS

This appendix summarizes additional equations developed to describe private sector dynamics in more detail.* These equations, which were developed using an alternative set of assumptions, have not been included in the Automated Econometric Model. In summary, in this appendix we sketch one approach to use in refining the model further.

Private sector incentives

The revised equations are intended to yield an annual average unit cost per line of code delivered. Synthetic though it is, such an average unit cost is the primary motivation for private sector firms. As the unit cost decreases, it demonstrates and quantifies the economic benefits obtainable when engineers, systems analysts and programmers working with the software tools in an environment, produce software of equivalent quality at lower unit cost. (Incidentally, this sort of replacement of labor with machinery is referred to as a "Cobb-Douglas" function).

The resulting costs, expressed in percentages, will look like this:

	Unit Cost per Line of Code	
DIRECT COST FOR ANALYSTS/PROGRAMMERS	$ 1.0000	
GENERAL OVERHEAD, AT 50%	.5000	
CONTROL STRATEGY	.0100	(example)
MANAGEMENT FUNCTIONS	.8800	
UNIT COSTS	$ 2.3900	

Assumptions

Assumptions underlying this appendix, which differ from those presented in the report, are as follows:

a. The primary driver is total annual USAF MCCR software WORKLOAD(n) [the quantity of software to be developed and the amount to be modified each year, expressed in 1000 lines of code]. This determines the number of potential users of environments.

b. Potential users are assumed to accept a new "challenger" ENVIRONMENT (i) or to conduct "Business as Usual," using their present practices, procedures, and software tools ENVIRONMENT (bau). To reflect time delays in procurement, installation, and training, the number of "challenger" environments (i,n) brought into service each year is limited using a "Pearl Curve" equation.

* Adapted from Automated Econometric Model User's Guide, June 20, 1986, pp. 3-1 to 3-7. The User's Guide presents a full description of the equations and their implementation using LOTUS 1-2-3.

c. A time delay is considered, between funding of initial R&D and beginning of benefits. During this period costs are incurred for the environment, but no benefits are realized.

d. Finite (though perhaps not measurable) costs are assumed for:

* Staff, both Government and contractor, to implement the nine Control Strategies.

* Continuing improvement in the capabilities of all MCCR software development/ support environments. An important vendor argument against use of any GFE/environment is that any required standard would inevitably result in degraded product quality or productivity. Because of the unprecedented rapidity of technological change in the computing fields, which renders one year's standard the next year's obsolescence, this argument could be valid.

* Contractors' strategy and motivations, based on vendors' business concerns, such as market shares and vested interests in their own software tools, technology, and products.

* "off-books" costs and benefits such as the effects of depreciation/amortization, and "opportunity costs" that cannot be ignored by contractors.

Equations

As before, the equations determine costs, benefits, and net benefits involved with implementation of a selected environment (i) using a selected control strategy (j). The environment-control strategy pair is referred to as a scenario (i,j). The model tracks each scenario by year (n) for the period 1986-1995. Thus equations may contain the three subscripts (i,j,n).

The model consists of workload equations, cost equations, and benefit equations. Net benefits of implementing the scenario (i,j,n) represent the sum of benefits less the sum of costs over the years in the period modeled.

WORKLOAD EQUATIONS

Eqn a: **WKLOADAF(n) = GOVTWKLD(n) + CNTRWKLD(n)**, where:

WKLOADAF(n) is TOTAL USAF MCCR Software Workload for year (n), in 1000 lines of delivered code. It can be estimated from AFSC data, or from the E.I.A. [Electronic Industries Association] 1986 projection.

GOVTWKLD(n) is TOTAL USAF MCCR Software Workload served by Government employees.

CNTRWKLD(n) is TOTAL USAF MCCR Software Workload served by Contractors and their employees.

Eqn b: **REQDPERS(n) = WKLOADAF(n)/PLOC(n)**, where:

REQDPERS(n) is the number of direct labor personnel needed to service the workload in year (n);

WKLOADAF(n) is the workload to be serviced; and

PLOC(n) is the Production Lines of Code to be delivered per required direct labor person in year (n).

Eqn c: **ENVTS(i,n) = REQDPERS/(1 + a^{-b*n})**, where:

ENVTS(i,n) is the Maximum number of "challenger" environments (i) that can be operational in year (n);

a is "Pearl curve" constant [set at about 1000]; and

b is "Pearl curve" constant [set at about 0.7].

Eqn d: **ENVTS(bau,n) = [WKLOADAF(n) - (ENVTS(i,n) * PLOC(i,n))] / PLOC(bau,n)**,

where:

ENVTS(i,n) is the Quantity of "challenger" environments in use during year (n);

ENVTS(bau,n) is the quantity of current environments in use in year (n);

PLOC(i,n) is the Production Lines of Code to be delivered per required person in year (n).

COST EQUATIONS

Eqn e: **DLABCOST(n) = ACPY(n) * [ENVTS(bau,n) + ENVTS(i,n)]**, where:

DLABCOST(n) is the cost for direct labor in year (n); and

ACPY(n) is average (total billed) cost per [direct labor] environment user in year (n).

Eqn f: **INDCTPCT(i,n) = CONFIGPC(i,n) + QUALASSU(i,n) + DOCNSUPP(i,n) + TESTSUPP(i,n) + PROGSUPP(i,n) + TRAINSUP(i,n) + CTLSTRAT(i,n)**, where:

CONFIGPC(i,n) is percent of direct labor cost for configuration management for the selected environment (i) in year (n);

QUALASSU(i,n) is percent of direct labor cost for quality assurance for the selected environment (i) for year (n);

DOCNSUPP(i,n) is percent of direct labor cost for supporting documentation for selected environment (i) in year (n);

TESTSUPP(i,n) is percent of direct labor cost for test support for the selected environment (i) in year (n);

PROGSUPP(i,n) is percent of direct labor cost for programming support for selected environment (i) in year (n);

TRAINSUPP(i,n) is percent of direct labor cost for training cost associated with environment (i) in year (n); and

CTLSTRAT(j,n) is percent of direct labor cost required to develop, oversee, and supervise implementation of the control strategy (j) in year (n).

Note that the percentage cost to implement varies by the control strategy chosen.

Eqn g: **OC(i,j,n) = [DLABCOST(n) * INDCTPCT(i,n)] + CENTCOST(i,n) + [SITECOST(i,n) * [CONSIT(n) + GOVSIT(n)] + DEPREC(i,n) + [ENVTS(i,n) * VBLCOST(i,n)]**

where:

CENTCOST(i,n) = Annual cost of central control facility;

SITECOST(i,n) = Annual cost of each site at which environment (i) is in use;

VBLCOST(i,n) = Variable costs [power, maintenance, light, air conditioning, data communication costs, etc.] for each separate environment (i) installed during year (n); and

DEPREC(i,n) = Annual depreciation charges for environments (i).

BENEFIT EQUATIONS

Eqn h. **UCOSTLOC(i,j,n) = COST(i,j,n) / WKLOADAF(n)**, where:

UCOSTLOC(i,j,n)	is Unit cost per line of code delivered;
COST(i,j,n)	is TOTAL COST for implementing scenario (i,j) for year (n); and
WKLOADAF(n)	is TOTAL USAF MCCR Software Workload for year (n), in 1000 lines of delivered code.

APPENDIX C

CAUSAL CHAIN USED AS BASIS FOR MODEL

This material is adapted from [Werl85], pp. V-5 to V-9.

ASSUMPTIONS AND CAUSE-EFFECT CHAIN LEADING TO SUCCESSFUL IMPLEMENTATION OF THE GFE APPROACH

Introduction

Post mortem reviews of complex programs often reveal that the assumptions made early in a program were incomplete or incorrect. For a program as complex as the GFE/Environment we must verify, to the extent possible, the reasonableness of the assumptions and of their assumed results. The "causal chain" approach, a graphic research tool, is useful for analyzing complex programs that involve economic, organizational, and technical assumptions.

The causal chain is used most for analyzing projects that require successful negotiation of a series of steps before they can be completed. The chain is used because it helps identify and make explicit the assumptions made at each step, and the reasonableness of the results expected from each successive action in the chain.

Implementing the GFE/Environment

Figure C-1 shows the chain of assumptions and cause-effect relationships for the GFE/Environment. It begins with "Generic standards are valuable" and ends with "Lower cost to DoD for new software." The three rows of boxes crossing the figure show the assumptions and cause-effect relationships for: (1) use of a single standard programming language (top row); (2) use of a GFE/Environment (middle row); and (3) the resulting effect on a defense system that contains hardware, software, facilities, data, and people (bottom row).

Quantifying assumptions. The four shaded boxes at the left, and the one near the middle of the figure, represent basic assumptions that are made (sometimes implicitly) and seldom questioned. The three dashed boxes at the right represent results desired from the project. Throughout the figure, each pair of boxes connected by an arrow represents a cause-effect transaction that is assumed to be effective.

We showed 27 numbered boxes on figure C-1. We can quantify or measure at points associated with each of the 27 boxes, then include the measurements in the equations of an econometric model. In Table C-1 we indicate the box numbers, and show the type of information required by the econometric model for those points. We also show Technion International's estimates of expected ranges of data.

POINTS AT WHICH ADDITIONAL DATA ARE NEEDED

This section, keyed to Figures C-1 and C-2, and Table C-1, details the data required and gives a range of probable values. Figure C-2 shows excerpts from the flow of Figure C-1, with more emphasis on blocks affecting the implementation of a GFE/Environment. Table C-1 shows specific data points and indicates ranges of values expected.

Quantification needed

Quantification is needed at the following points in Figures C-1 and C-2.

Language:

- What is the improvement in programmer productivity, both for development and for subsequent enhancement/maintenance of software, associated with use of one standard programming language? (Box 3)

- What is the incremental expense of training programmers in the standard language, to the skill level at which they can implement the language's special features in their work? (Box 5).

- What are the net benefits to programmers of using the language? (Box 6)

- What are the net benefits to contractors from having their programmers use the language? (Box 7).

- To what degree are reliability and maintainability enhanced by having software written in the standard language? (Box 8).

Environment:

- What are the benefits of a standard integrated automated environment, in turns of improved productivity in development, reduced time to complete testing, and improved productivity in subsequent enhancement and maintenance of software? (Box 12).

- What will be the acquisition cost of a GFE/Environment [i.e., costs for developing requirements, acquiring a GFE/Environment, and providing multiple copies as GFE to contractors]? (Box 13).

- What will be the cost of training programmers in use of the GFE/Environment (Box 14).

- What are the incremental costs and benefits of having contractors use the GFE/Environment, instead of their customary software production facilities? (Box 15).

- How much lower will be the cost to DoD for development of NEW software? (Box 9).

- How much lower will be the cost to DoD for subsequent enhancement and continuing development of software? (Box 10).

System containing the software

- What is the likelihood that characteristics of the defense system (which drive the software development and maintenance efforts) are compatible with the design of the GFE/Environment and the project in which it is used [e.g., system budget, schedule, required reliability, complexity, and volatility of requirements on software]? (Boxes 19 and 20).

- What will be the effect of the resulting software on reliability and maintainability of the defense system over its life cycle? (Box 22).

- What will be the effect of the resulting software on the performance of the defense system? (Box 27).

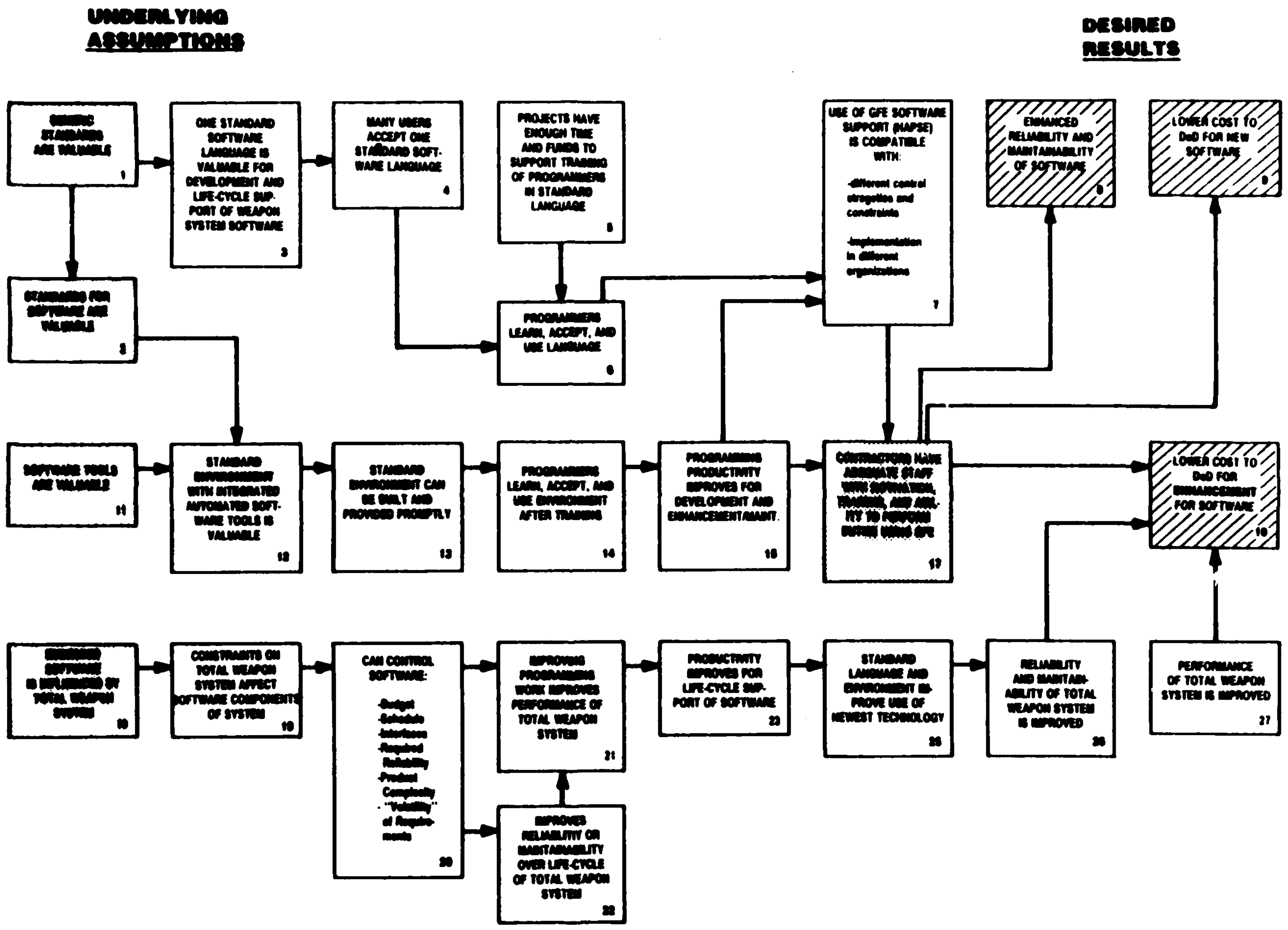
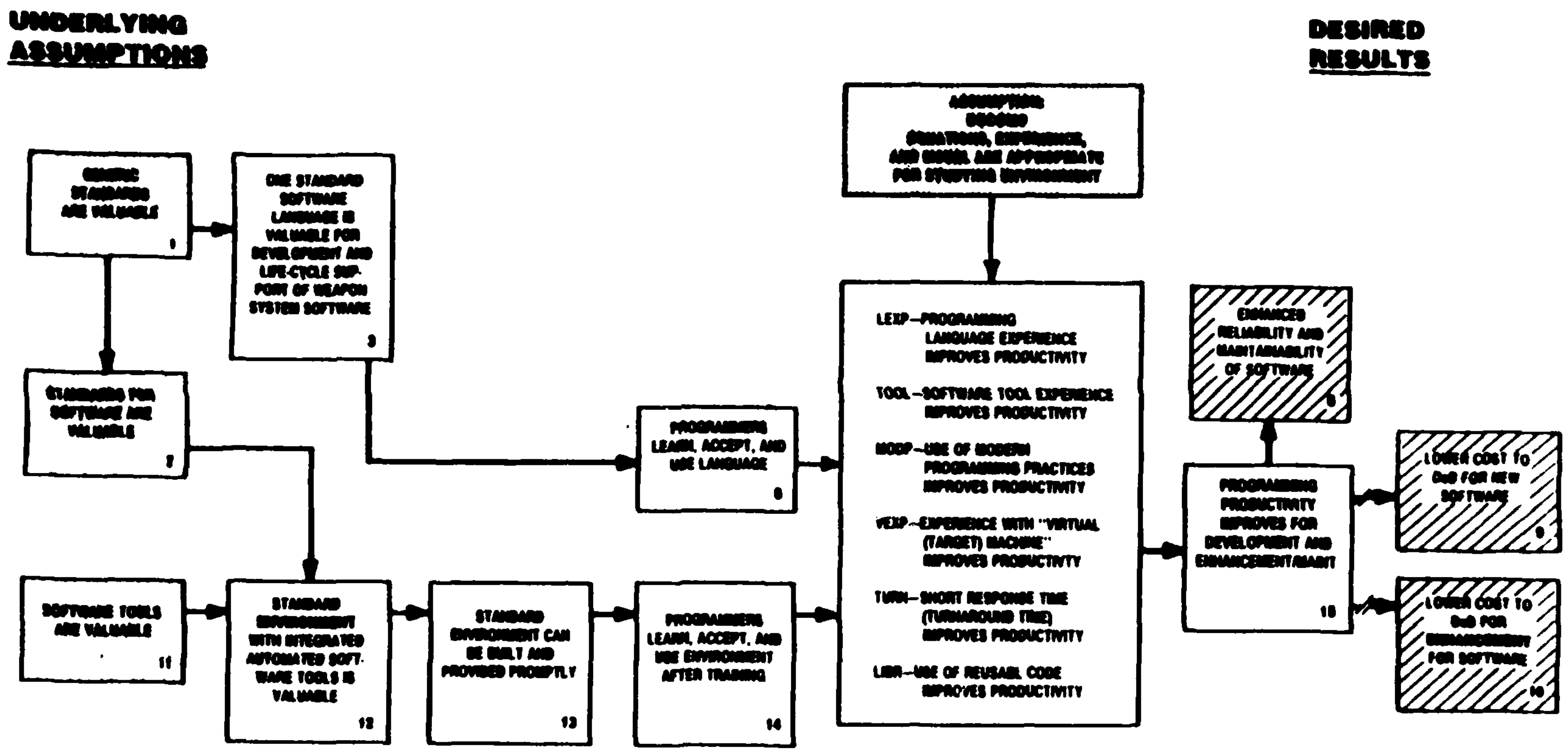

UNDERLYING ASSUMPTIONS
DESIRED RESULTS
GENERIC STANDARDS ARE VALUABLE 1
ONE STANDARD SOFTWARE LANGUAGE IS VALUABLE FOR DEVELOPMENT AND LIFE-CYCLE SUPPORT OF WEAPON SYSTEM SOFTWARE 3
MANY USERS ACCEPT ONE STANDARD SOFTWARE LANGUAGE 4
PROJECTS HAVE ENOUGH TIME AND FUNDS TO SUPPORT TRAINING OF PROGRAMMERS IN STANDARD LANGUAGE 5
USE OF GFE SOFTWARE SUPPORT (HAPSE) IS COMPATIBLE WITH: -different control strategies and constraints -implementation in different organizations 7
ENHANCED RELIABILITY AND MAINTAINABILITY OF SOFTWARE 8
LOWER COST TO DoD FOR NEW SOFTWARE 9
STANDARDS FOR SOFTWARE ARE VALUABLE 2
PROGRAMMERS LEARN, ACCEPT, AND USE LANGUAGE 6
SOFTWARE TOOLS ARE VALUABLE 11
STANDARD ENVIRONMENT WITH INTEGRATED AUTOMATED SOFTWARE TOOLS IS VALUABLE 12
STANDARD ENVIRONMENT CAN BE BUILT AND PROVIDED PROMPTLY 13
PROGRAMMERS LEARN, ACCEPT, AND USE ENVIRONMENT AFTER TRAINING 14
PROGRAMMING PRODUCTIVITY IMPROVES FOR DEVELOPMENT AND ENHANCEMENT/MAINT. 15
CONTRACTORS HAVE ADEQUATE STAFF WITH INFORMATION, TRAINING, AND ABILITY TO PERFORM DUTIES USING GFE 17
LOWER COST TO DoD FOR ENHANCEMENT FOR SOFTWARE 10
EMBEDDED SOFTWARE IS INFLUENCED BY TOTAL WEAPON SYSTEM 18
CONSTRAINTS ON TOTAL WEAPON SYSTEM AFFECT SOFTWARE COMPONENTS OF SYSTEM 19
CAN CONTROL SOFTWARE: -Budget -Schedule -Interfaces -Required Reliability -Product Complexity -"Volatility" of Requirements 20
IMPROVING PROGRAMMING WORK IMPROVES PERFORMANCE OF TOTAL WEAPON SYSTEM 21
IMPROVES RELIABILITY OR MAINTAINABILITY OVER LIFE-CYCLE OF TOTAL WEAPON SYSTEM 22
PRODUCTIVITY IMPROVES FOR LIFE-CYCLE SUPPORT OF SOFTWARE 23
STANDARD LANGUAGE AND ENVIRONMENT IMPROVE USE OF NEWEST TECHNOLOGY 25
RELIABILITY AND MAINTAINABILITY OF TOTAL WEAPON SYSTEM IS IMPROVED 26
PERFORMANCE OF TOTAL WEAPON SYSTEM IS IMPROVED 27

Figure C-1 - Assumptions and Causal Chain -- GFE/Environment.



UNDERLYING ASSUMPTIONS
DESIRED RESULTS
GENERIC STANDARDS ARE VALUABLE 1
ONE STANDARD SOFTWARE LANGUAGE IS VALUABLE FOR DEVELOPMENT AND LIFE-CYCLE SUPPORT OF WEAPON SYSTEM SOFTWARE 3
STANDARDS FOR SOFTWARE ARE VALUABLE 7
SOFTWARE TOOLS ARE VALUABLE 11
STANDARD ENVIRONMENT WITH INTEGRATED AUTOMATED SOFTWARE TOOLS IS VALUABLE 12
STANDARD ENVIRONMENT CAN BE BUILT AND PROVIDED PROMPTLY 13
PROGRAMMERS LEARN, ACCEPT, AND USE ENVIRONMENT AFTER TRAINING 14
PROGRAMMERS LEARN, ACCEPT, AND USE LANGUAGE 6
ASSUMPTION: COCOMO (EQUATIONS, EXPERIENCE, AND MODEL) ARE APPROPRIATE FOR STUDYING ENVIRONMENT
LEXP—PROGRAMMING LANGUAGE EXPERIENCE IMPROVES PRODUCTIVITY
TOOL—SOFTWARE TOOL EXPERIENCE IMPROVES PRODUCTIVITY
MODP—USE OF MODERN PROGRAMMING PRACTICES IMPROVES PRODUCTIVITY
VEXP—EXPERIENCE WITH "VIRTUAL (TARGET) MACHINE" IMPROVES PRODUCTIVITY
TURN—SHORT RESPONSE TIME (TURNAROUND TIME) IMPROVES PRODUCTIVITY
LIBR—USE OF REUSABL CODE IMPROVES PRODUCTIVITY
PROGRAMMING PRODUCTIVITY IMPROVES FOR DEVELOPMENT AND ENHANCEMENT/MAINT 18
ENHANCED RELIABILITY AND MAINTAINABILITY OF SOFTWARE 5
LOWER COST TO DoD FOR NEW SOFTWARE 9
LOWER COST TO DoD FOR ENHANCEMENT FOR SOFTWARE 10

Figure C-2. Causal Chain - Programming Support Environment.

Table C-1. Data Required by Econometric Model.

Figs C-1, -2 Box number:	Data Required	Range of probable values
(n/a)	GFE/Environment is improved, and overcomes pressures toward obsolescence.	GFE/Environment continues to provide productivity improvements amount to at least ten percent per year after its deployment.
(5)	Additional funding needed by contractors to train their people in using GFE/Environment.	$1000-$5000 per programmer to be trained (one to five weeks each)
		"Learning curve" effects may include: initial loss of productivity, followed by shift to more favorable GFE/Environment learning curve, with long-term gain.
(4), (12)	Productivity is raised by language and software tools used in GFE/Environment.	Ada language produces from eight to ten machine-language instructions per Ada language instruction.
(6), (14)	Programmers learn, accept, and use GFE/Environment after training.	Set of integrated, automated software tools helps programmers produce better products and becomes part of their "normal" toolkit.
(13)	Standard environment can be built and provided promptly.	Yes, within four years. Acquisition cost depends on acquisition strategy. Additional annual direct support cost will also be significant.
(15)	Productivity improves for <u>initial development</u> as well as for post-deployment enhancement/maintenance of software.	Productivity increases by factor of two to four.

(continued)

Table C-1, concluded

(17)	Contractors have adequate staff, with motivation, training, and ability to perform their work using the GFE/Environment.	From 25 to 50 percent of contractors' project staff have adequate motivation, training, and ability to use the GFE/Environment.
(25)	GFE/Environment aids in transition and use of latest technology on AFSC MCCR projects.	Range of values, from plus 50 percent (net help) to minus 25 percent (net interference) with transition.
(26), (8)	GFE/Environment use gives improved reliability, maintainability, and effectiveness of defense system.	Range of values, from zero to doubling of R-M-A with accompanying decrease in post-deployment support costs.
(9)	GFE/Environment lowers costs for developing new software.	Two to fourfold increase in productivity, with greater reliability and maintainability of new software products.
(10)	GFE/Environment lowers time and costs for post-deployment modifications of fielded software.	Doubled productivity in post-deployment enhancement/continued development of operational software.

Estimates for data range were made by Technion International, and are based on a wide variety of published and unpublished reports. However, data in the right column should be used for no purpose other than to plan data collection efforts.

APPENDIX D

EXCERPTS FROM [WERL86][1]

[Can USAF improve productivity by more than ten percent each year?]

. . . Contractors are tripling this rate now, so we believe USAF can do the same. This conviction underlies the rest of this report.

CURRENT COST ESTIMATING TECHNIQUES

We learned during our visits [to six AFPROs and software contractors' sites] that contractors develop cost estimates using their normal internal accounting records. Contractors use parametric cost estimating models primarily as "sanity checks."

Estimating often a Sales Function.

We spent only a few hours with contractors' cost estimating staff. However, it seemed to us that contractors' software estimating procedures are intended to justify estimates rather than to control costs. We noted that in contractors' organizations, cost estimating and pricing are often sales/marketing functions.

Contractors' estimates rely on data from their accounting records

Contractors' pricing systems are based on their "local" accounting records. The typical cost estimate is assembled using the "bottom-up" approach, with each supervisor estimating the time required by the portion of the work for which his organizational unit will be responsible. The total cost is arrived at by summing these individual estimates. This method has both strengths and weaknesses.

Strengths. Estimates are prepared by those individuals most familiar with the organizational units that will do the work. Thus, if supervisors have recent experience with the products and technologies to be used, their estimates can be very good.

Weaknesses. When preparing competitive proposals, contractors' pricers usually think first in terms of "pricing to win," independent of the actual difficulty of the work to be done. Thus estimates reflect: (a) how much funding they think USAF has

[1] Werling, R., "Data Collection System for Estimating Software Development Cost." Report of research conducted for USAF Business Research Management Center, AFBRMC/RDCB, Wright-Patterson AFB, OH. September 1986. This research was supported under contract number F 33615-85-C-5123.

available for the product; (b) their probable competitors' related products, capabilities, and experience; (c) the extent to which "buying in" may be appropriate to their corporate plans; (d) their own units' profit/loss position; and (e) the level of their firms' backlogs. In general, these act as incentives to underestimate the time and resources needed.

. . . Because the work is done by many estimators, their situations and motivations may differ substantially. Managers higher in the organization normally try to overcome this problem by reviewing aggregated estimates as the process proceeds. On balance, these probably act to over-estimate time and resource needs.

Software not estimated separately.

Contractors rarely estimate and price software products separately. In accordance with current regulations, they design the system work breakdown structure (WBS) around hardware deliverable components. Their cost reporting and accounting systems then treat software products simply as components of the hardware deliverables. As a rule, then, contractors collect only minimal detail on software costs.

MANAGING PRODUCTIVITY FOR MCCR SOFTWARE DEVELOPMENT AND SUPPORT

USAF not benefiting from improved software productivity

For a combination of reasons, USAF is not getting the benefit of the [10-15 percent] annual improvement in software productivity observed in the industry. During our visits to AFPROs, we were startled to find that contractors' present procedures for estimating software costs assume no corresponding growth in productivity. Instead of the annual improvement of 10-20 percent we expected, contractors' estimates typically assume that development procedures and conditions will be the same as those used several years earlier. In effect, this leads to zero growth in productivity, and to higher than appropriate costs to USAF. Yet, such benefits are clearly available to USAF [Bita84], [Boeh81], [Boeh84], [Druf82], [Mats81], [McNa85].

Software industry achieving productivity gains

Figure [I-1], from an earlier analysis by Technion International, shows annual productivity increases exceeding 20 percent [Werl85, p. IV-15]. Our statistical analysis of software projects similar to MCCR showed that, when everything else is held constant, annual productivity grew at more than 20 percent during the 1970s. Although not yet recognized in management literature, the phenomenon was acknowledged in discussions with contractors. One contractor had observed a doubling of his organization's software productivity during the prior seven years [10.4% compounded], and projected an additional doubling in the next five years [14.9% compounded]. Another's estimate was in the same range. The program described below is intended to help AFPROs approach these rates in negotiating similar annual improvements.

Development technology has already raised productivity. Our research shows that a number of contractors have systematic MCCR software productivity improvement programs. Software technology breakthroughs have been frequent and varied. During the decade since 1975: (a) interactive programming has replaced more time-consuming manual processes; (b) computer speeds and memory capaacities have improved exponentially; (c) compilers have become more effective, faster, and more helpful to users; and (d) "modern programming practices" and "software tools" help throughout the entire software development/support cycle. Taken together, these advances have transformed the way production software engineers do their work. The growth in improvement is continuing, helped by recent DoD-sponsored efforts, such as the ADA*, STARS, and SEI, which promise ongoing forces to improve the process.

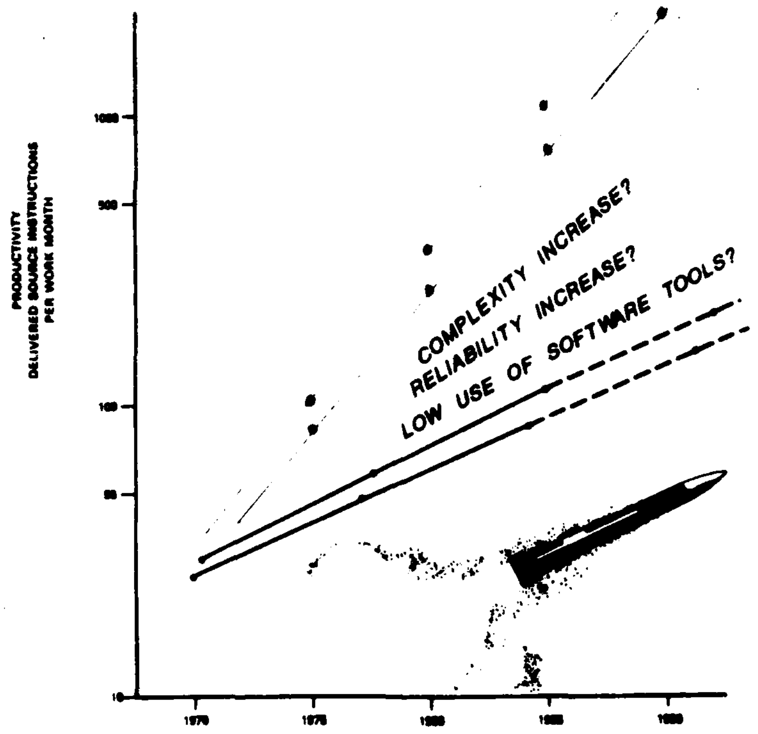
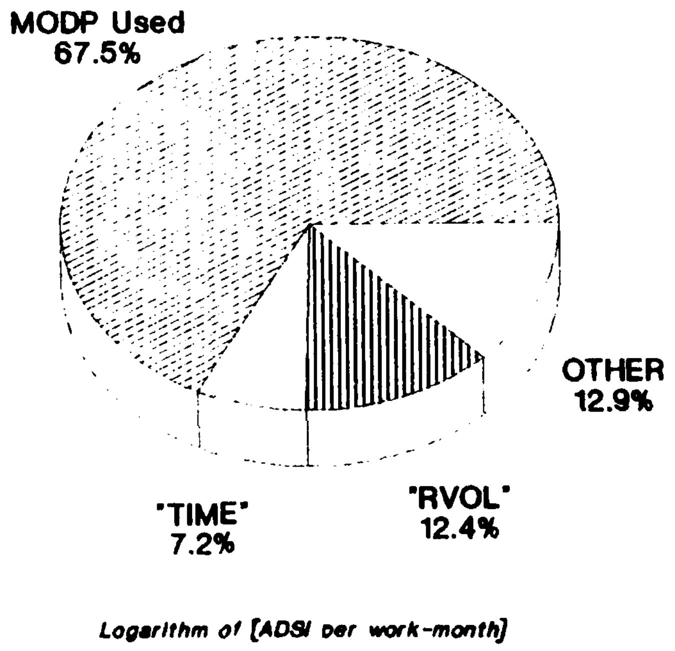


Figure [D-1]. Where has the productivity gone?

The way to go. Our statistical analysis of 34 MCCR-like projects from the COCOMO project data base [Boeh81, pp. 496-7] showed that a handful of cost drivers account for the lion's share of productivity improvements. For example, regression analysis revealed that during the 1970s three cost drivers explained 87 percent of the variability in productivity. Figure [D-2] relates variability in the logarithm of productivity [actual delivered source instructions per work-month] to cost drivers. It shows that three drivers (Use of Modern programming practices, Time constraint in target computer, and Volatility of requirements) were able to explain most of the variability in productivity found for the sample of 34 projects.



Sources: Boehm 1981, Technion Int'l

Figure D-2. Productivity cost drivers.

Summary

We have seen that software productivity improved, along with technological advances, during the 1970s. Because a handful of cost drivers exerts inordinate influence over project cost, duration, and productivity, USAF can find levers to improve them. Why has USAF not received the benefit of these technological improvements? There are many reasons and many more conjectures. Some of the reasons can be addressed by AFPROs and SPOs.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release: Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) BRMC-85-5165-1
6a. NAME OF PERFORMING ORGANIZATION Technion International, Inc. — 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Air Force Business Research Management Ctr AFBRMC(RDCB)
6c. ADDRESS *(City, State and ZIP Code)* P.O. Box 417 Wilmington, DE 19899	7b. ADDRESS *(City, State and ZIP Code)* Wright-Patterson AFB OH 45433
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFBRMC — 8b. OFFICE SYMBOL *(If applicable)* RDCB	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-5165
8c. ADDRESS *(City, State and ZIP Code)* Area B, Bldg 125 Wright-Patterson AFB OH 45433	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
71113	0	08	0

11. TITLE *(Include Security Classification)*
Cost Effectiveness Trade-Offs in S/W Support Environment

12. PERSONAL AUTHOR(S)
Richard Werling

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
Final	FROM 850929 TO 860930	860930	114

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.
5.3	14.1	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Econometrics, equations, integrated systems, models,
Productivity standards, cost effectiveness, control computer

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The principal objective of the study was to develop a multi-equation simulation model for determining the quantitative benefits of implementing strategies for controlling integrated, automated software support environments. The model describes support activities for mission-critical computer resources (MCCR). It is an econometric model which describes the cost and benefits of using Ada-based computer software development/support environments and permits comparison of these environments. Four types of environments are discussed: framing, programing, general and methodology-specific.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER *(Include Area Code)*	22c. OFFICE SYMBOL
Capt Edward C. Mitchell	(513) 255-6221	RDCB

END
8-87
DTIC