Order Number 1355627

Computer-aided software engineering (CASE) and productivity

Garro, Ligia Maria, M.S.

The American University, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

COMPUTER-AIDED SOFTWARE ENGINEERING (CASE) AND PRODUCTIVITY

by

Ligia M. Garro

Submitted to the

Faculty of the College of Arts and Sciences

of The American University

in Partial Fulfillment of

the Requirements for the Degree

of

Master of Science

in

Information Systems

Signatures of Committee:

Chair: Robert L. Crosslin

Gene McGuire

Dean of the College

5-5-93

Date

1993

The American University

Washington, D.C. 20016

COMPUTER-AIDED SOFTWARE ENGINEERING (CASE)

AND PRODUCTIVITY

BY

Ligia M. Garro

ABSTRACT

Much has been said of the benefits obtained from using Computer-Aided Software Engineering (CASE) technology in the development of information systems (IS). Do these benefits occur?.

The purpose of this thesis is to study how organizations are measuring productivity improvements from CASE. Hypotheses are established to conduct a survey in organizations where CASE is being used.

First, a literature review is conducted to understand this technology, and to study what should be considered to ensure its successful implementation in an organization. Second, different definitions of productivity and several examples of measurements are presented as guidelines to measure IS activities. Success stories of CASE implementation complete this bibliographical research.

An in-depth study, using questionnaires and interviews, was conducted in three multinational organizations based in the Washington, D.C. area. The results show that in these

organizations, the introduction of CASE formed part of the IS strategic plan, and that as a result, the systems development life cycle has changed. However, a definition for productivity did not exist and there were no measures for capturing it. There was also a lack of historical data relating to IS activities, which does not allow any comparison between IS activities before CASE and activities as they are carried out now.

As a conclusion, it becomes clear that some aspects have been forgotten or underestimated when implementing CASE. Also, IS management has failed to evaluate how this technology has affected the development of systems. Top management's perceptions of IS activities might have influenced this situation by not recognizing the need to assign resources to IS departments to perform this evaluation. In spite of this situation, the results of this thesis indicate that it is worth investing in CASE as a means to facilitate and standardize the way systems are developed.

ACKNOWLEDGMENTS

This thesis would not have been possible without the invaluable cooperation of a number of information systems professionals, who were willing to take time from their busy schedules to answer the questionnaire, and give me the opportunity to interview them. My special thanks to all of them.

I am also extremely grateful to my thesis committee for its support and advice during the development of my thesis.

TABLE OF CONTENTS

LIST OF TABLES

Table

CHAPTER I

INTRODUCTION

For years information systems (IS) professionals have been developing applications to make users work less burdensome, through the automation of many of their tasks. However, IS professionals have found themselves without automated tools to facilitate their job.

Computer-Aided Software Engineering (CASE) tools have been available to support the systems' development process since the early 1980's. CASE tools have also been proclaimed as a cure to alleviate IS problems, and to improve IS productivity. Because so much has been said about how this technology improves productivity, it is important to study the relationship between CASE and productivity. This is precisely the purpose of this thesis.

A literature review was conducted to learn about this technology, the factors that should be considered in order to assure its successful introduction in an organization, some definitions for productivity, and measurements which can be used to evaluate the effectiveness and efficiency of IS activities. Also, some examples of successful CASE implementation stories are presented to illustrate the

benefits of CASE.

The literature review was also the basis to conduct a survey to study how specific organizations are dealing with CASE regarding productivity. Three multinational organizations in the Washington, D.C. area were comprehensively studied. IS professionals working in these organizations answered a questionnaire designed for the purpose of establishing the productivity of CASE. Interviews were also conducted not only with the questionnaire's respondents, but also with IS chiefs and IS auditors from the respondents' organizations.

The results of the in-depth study of the organizations selected show that CASE introduction was part of the IS strategic plan, and that the systems' development process has changed to some extent because of it. However, in these organizations there is no definition for productivity and no measures for it. The lack of historical data on IS activities makes it impossible to answer whether CASE technology is improving systems development. Nevertheless, there is a feeling that systems' quality is being improved using CASE tools.

The study leads to the conclusion that even when an organization acquires CASE for the right reasons, some factors are ignored or underestimated when trying to obtain all the potential benefits brought by this technology. Also, there is a lack of management commitment in assigning

resources to evaluate IS activities, in particular, the effect of CASE on the systems' development process.

CHAPTER II

CASE TECHNOLOGY

Since the creation of the first computer and the first programming language, the IS field has changed dramatically. New and better machines, new and better programming languages, new IS development methodologies, and more knowledgeable users have led us to an increasingly formal activity; that of building systems. But not every aspect in this arena has matured at the same rapid pace.

"No shoes for the shoemaker's children. " This popular adage illustrates what has been happening to professionals developing IS since the very beginning. On one hand, they have been, and still are, helping users in their tasks by providing them with tools that facilitate them, and allow them to perform their jobs more efficiently. On the other hand, IS professionals have found themselves without automated tools to improve their job.

Would not an IS professional prefer to have an automated tool to help himself/herself in doing his/her job? Would not it be more convenient to have a tool to create structured diagrams, check inconsistencies, ease modifications, produce some documentation, and generate

code?.

One of the wishes of most IS managers has been to have a tool for improving productivity, for building quality systems and for reducing and facilitating maintenance work. Since the early 1980's, when the first Computer Aided Software Engineering (CASE) tools were introduced, new heights have been reached with regard to these objectives.

Background of CASE

Carma McClure[1] defines CASE simply as software automation. For her, the introduction of computer-aided documentation and diagraming techniques in the early 1980's marks the beginning of CASE technology. These PC-based tools attempted to support systems development in analyzing and designing systems, and in documenting these tasks.

McClure[2] points out that later, during the mid-1980's, CASE tools were improved to provide automatic design, analysis and checking, and to store structured diagrams in automated design libraries (dictionaries, repositories, encyclopedias). In the late 1980's, automatic code generation from design specifications, and the link between design automation and program automation were characteristics of CASE tools. Fourth-generation languages

---

[1] McClure, Carma. CASE is Software Automation. Englewood Cliffs, New Jersey: Prentice Hall. 1989. pp. 8-14.

[2] Ibid.

(4GL) and code generators, mainly based on mainframes, helped in providing these characteristics.

In 1989, another definition for CASE was offered by Gibson[3], who defined CASE as a comprehensive philosophy for modeling a business, its activities, and the development of IS. The key concept here is the use of the computer as a development tool. CASE tools allow IS professionals to build models that describe the business the company is in, its corporate planning and to document systems development from the very beginning, planning, to the very end, construction.

Today, there is no doubt that systems' development is becoming a much more complex activity. Therefore, a disciplined and standardized method is needed to manage it. This fact lead us to another definition[4] for CASE, which calls primarily for an engineering approach for systems development. Following this concept, in May 1992, CASE was defined by the International Workshop on Computer-Aided Software Engineering[5] (IWCASE) to include tools and methods supporting an engineering approach to software development at all stages of the process.

---

[3] Gibson, Michael L. "The CASE Philosophy," Byte. April 1989. p. 209.

[4] QED Information Systems, Inc. CASE: The Potential and The Pitfalls. Wellesley, Massachusetts. 1989. p. 15.

[5] Forte, Gene and Norman, Ronald J. "A Self-Assessment by the Software Engineering Community," Communications of the ACM. May 1992. p. 29.

The idea behind all these definitions is that CASE tools emphasize structured methods, and a disciplined approach toward building systems. Different CASE tools are based on different development methodologies such as Yourdon's structured analysis and design, or Jackson's structured design, or Martin's information engineering. Broadly speaking, a CASE tool forces the IS professional to use a methodology and to proceed in a disciplined fashion when developing systems.

The definitions presented above are not in conflict. In fact, they underline the aspect that CASE is not only a number of tools per se, but also it is a new way of developing systems. As a whole, the systems' development life cycle (SDLC) is then modified by the introduction of these tools, which support the automation of systems development and maintenance tasks. In this respect, the software development process becomes very interactive since many CASE tools are workstation-based. Also, checking of errors and inconsistencies can be done at the very first phases of the systems' development cycle.

CASE Tools

Although CASE tools are relatively new, they have evolved through the years. McClure[6] states that the definition for a CASE tool has broadened and that today a CASE tool is any software tool that provides automated support for project management activities, systems development, or systems maintenance. These tools are characterized by:

1. supporting a dedicated, personal computing environment,
2. using graphics for system specifications and for documentation purposes,
3. linking all phases of SDLC,
4. capturing and linking all information about a system throughout its life cycle, and
5. using artificial intelligence to routinely perform system development and maintenance tasks automatically.

Today several CASE tools are available. They can be grouped in different ways. As an example, one classification[7] is based on the stages of the SDLC they support:

Group 1: Tools supporting requirements definition and

---

[6] McClure, Carma. "The CASE Experience," Byte. April 1989. pp. 235-236.

[7] QED Information Sciences, Inc. CASE: The Potential and The Pitfalls. Wellesley, Massachusetts. 1989. pp. 553-565.

structured engineering.

These tools support several different forms of structured engineering methodologies. They may contain structured techniques that support one or more of the structured methodologies, such as Yourdon-DeMarco, Gane-Sarson, Warnier-Orr, Jackson, and Constantine. These tools include support for drawing entity-relationship diagrams, process decomposition diagrams, process dependency diagrams, data flow diagrams, data structure diagrams, and process action diagrams.

Tools of this group help in business system design, process modeling, or data modeling. Examples are DesignAid (Nastec Corporation), Excelerator (Index Technology, Inc.), and Analyst Toolkit (Yourdon Software).

Group 2: Tools that provide systems analysis and design aids.

These tools support specific diagram types. They enforce procedures of specific methodologies, maintain an integrated design repository, ensure the consistency of the system database, and generate system documentation automatically. Examples are Design/1 (Arthur Andersen), MAESTRO (Softlab Systems, Inc.), and TeamWork (Cadre Technologies).

Group 3: Tools that generate application code.

These tools are intended to generate executable code automatically from pictorial system specifications.

Examples are APS (Sage Systems, Inc.), GAMMA (KnowledgeWare, Inc.), and TRANSFORM (Transform Logic Corporation).

Group 4: Tools that support a full software engineering life cycle.

These tools have all the functions mentioned above, and include an information repository to maintain enterprise models, data models, and process models. They are a set of tightly integrated, formal techniques. The life cycle process is based on a top-down view of the company's business and IS strategies.

Examples are Information Engineering Facility (Texas Instruments), Information Engineering Workbench (KnowledgeWare, Inc.), and Application Factory (Cortex Corporation).

Group 5: Tools that generate correct specifications.

One of the CASE technology goals is to prove the logical correctness and completeness of systems specifications. There are some techniques that accomplish this goal such as program logic, expert systems heuristics, and associated rules of inference, and mathematical algorithms, which are incorporated into the tool. When specifications have been proved to be logically correct, they may be converted automatically into code. Information Engineering Workbench (KnowledgeWare, Inc.) and USE.IT (Higher Order Software, Inc.) are examples of these tools.

Components of CASE

Another commonly used way of classifying CASE is by life cycle usage. Gibson[8] proposes the breaking down of CASE into component parts, namely upper CASE, middle CASE, and Lower CASE, to facilitate its understanding.

Upper CASE.

This component supports models of the organization and its plans. Graphical designs are used to build enterprise models to find out the importance of organizational functions and how their activities affect the whole organization. Through the use of Upper CASE it is possible to obtain a better understanding of:

- corporate and departmental functions
- organizational goals
- operations and their effect on these goals
- timeliness and sequence of operations
- the allocation of resources to support operations
- problems affecting the organization
- the importance of information to the organization's success

Middle CASE.

It supports systems analysis and design. Corporate planning specifications are the input for designing IS specifications, which are enhanced using middle CASE tools.

---

[8] Gibson, Michael L. "The CASE Philosophy," Byte. April 1989. pp. 209-218.

Systems analysts use design specifications as the input for the development phase and user documentation. Most middle CASE tools use diagramming and dictionary components that work similar to those in upper CASE. A combination of components help to automate known systems methodologies used by systems analysts.

Some of the benefits obtained with the use of middle CASE include:

- easier methods for changing system design
- help to systems analysts to clearly understand the problems and how to solve them
- a reduction in time needed for the project development life cycle
- a structure for storing the knowledge of systems analysts about the organizational functions and information needs. This knowledge is normally kept in the systems developers minds.
- facilitating Joint Application Design (JAD) sessions

Besides these benefits, most middle CASE tools have a prototyping facility that allows the user to check how the system will work through screens and reports for user interface. This is very important since users can evaluate the systems functions without actually having the system completely built.

Lower CASE.

This component supports physical system development. It involves customized coding for specialized processing. Lower CASE creates the actual development specifications used to generate programs and to provide user documentation.

According to Gibson[9], lower CASE contains a dictionary system to specify the characteristics of the real world entities being modeled. This dictionary is an active one, letting the systems developer enter specifications that describe and influence the development of the modeled object by providing criteria for its development as well as references to its attributes.

Examples[10] of lower CASE benefits are:

- reduction in the time required to develop a system due to easier generation of program code
- ease to modify systems because maintenance activities usually only involve changes to custom code
- reduction of clerical work because many development specifications are reusable
- generation of development and user documentation in different formats
- production of prototypes that function like stand-alone systems not requiring specialized training

[9] Ibid. p. 212.

[10] Ibid.

to use them.

The breakdown of CASE presented above has not been universally accepted. Other authors[11] prefer to talk of only two components: upper CASE and lower CASE; defining upper CASE as the combination of what was described earlier as upper CASE and middle CASE. The lower CASE definition remains the same. In any case, these are only attempts to classify CASE tools. What should be important to remember is the various functions of the tools and how they can be used during the systems development activities in an organization.

CASE Categories

As stated earlier, there are many CASE tools for a systems developer to choose from. Therefore, it is crucial to find out first what each tool does and what its functions are. In this respect, McClure[12] presents two basic categories for CASE tools, namely, toolkits and workbenches.

A toolkit is a set of integrated tools that supports one phase of the SDLC, or a task such as analysis, database and file design, program implementation or project management. Toolkits used for analysis and design activities, for example, include screen and report painting, simulation, prototyping, and error checking functions. Many

---

[11] QED Information Sciences, Inc. CASE: The Potential and The Pitfalls. Wellesley, Massachusetts. 1989. p.15.

[12] McClure, Carma. "The CASE Experience," Byte. April 1989. pp. 235-244.

of them run on personal computers; they also support different structured methodologies such as Yourdon-DeMarco or Martin's Information Engineering. Excelerator from Index Technology and Analyst/Designer Toolkit from Yourdon are examples of available tools in the market. There are also toolkits to support the development of real time structured methodologies such as Ward-Mellor. Macintosh platforms can be used for toolkits as well.

Conversely, a workbench is a collection of integrated tools to provide automated assistance during the whole SDLC. In order to select a CASE workbench it is necessary to match hardware, development methodologies, and systems that the workbench supports to the user's development style and systems requirements. Information Engineering Facility (IEF) from Texas Instruments, and Information Engineering Workbench (IEW) from KnowledgeWare are examples of CASE workbenches.

Integrated CASE

Gibson, Snyder, and Carr[13] present integrated CASE (ICASE) as supporting methodologies that incorporate information engineering principles. The information engineering term was popularized by James Martin[14]. It is

---

[13] Gibson, Michael L.; Snyder, Charles, and Carr, Houston H. "Why CASE Belongs in Strategic Business Management," Information Strategy. Winter 1991. p. 18.

[14] Case, Albert F. "Information Engineering and CASE Environments," The CASE Report. November 1987. p. 2.

defined[15] as "an interlocking set of automated techniques in which enterprise models, data models, and process models are integrated in a comprehensive knowledge base used to develop and maintain systems. "

The importance of the information engineering concept is that it comprises system planning and system development as complementary activities. It uses methods with common notations and nomenclature for both activities. It is in this context that ICASE fits. ICASE shares specifications across CASE components: upper, middle, and lower CASE. Specifications are also shared across corporate planning, systems analysis and design, and systems development diminishing only the level of abstraction.

As Burke[16] points out there is a need for software tools that address the entire SDLC. According to him, the need for integration of single-user, single-phase tools has brought about ICASE. Integration is seen here as the ability to incorporate tools from all phases of the SDLC to work as one.

What is important in a truly integrated CASE environment is the possibility of having better communication between corporate strategists, corporate planners and systems developers. A common set of tools,

---

[15] Brathwaite, Kenmore S. Applications Development Using CASE Tools. San Diego: California: Academic Press, Inc. 1990. p. 259.

[16] Burke, John P. "Though CASE," HP Professional. July 1991. p. 32.

operating on a common database and at different levels of abstraction, ensures a better understanding of what an organization needs, as well as what is being done to fulfil those needs and why.

Introduction of CASE into an organization

A number of authors in the field have studied the phenomenon of introducing CASE technology in the IS organizations. Chikofsky[17], for example, points out that the biggest payoff from CASE is its potential to help organizations achieve crucial improvements in the way systems are developed and in terms of their quality. Geller[18] introduces the idea that CASE tools can be useful only when people know how to use them correctly.

A successful introduction of CASE is also determined by the organizational culture. Resistance to change cannot be overlooked. Buying a CASE tool is relatively easy, but getting it to work properly within an organization may be a real challenge. A formal planning effort for the introduction of CASE technology is a must.

One of the milestones in this planning effort is the

---

[17] Chikofsky, Elliot J. "Making CASE Pay Off," CASE Directions. Vol. 1, No. 1. p. 16.

[18] Geller, Rob. "Structured for Success," Information Week. April 6, 1992. p. 70.

assessment of implementation costs. Huff[19] indicates that CASE implementation costs in the organization tend to be higher than many organizations initially estimate. The cost of the actual CASE tool is only a small part of the total implementation cost. To this respect Huff proposes key budgetary components in four areas:

- Technology: including processes, methods, hardware, software, standards and practices
- Organization: including such characteristics as culture, policies, procedures, and interaction among organizational units
- People: including skills, knowledge, workload, motivation, and moral
- Management

All implementation costs are important when planning the introduction of CASE. However, organizational culture stands out as a crucial factor. What people expect from CASE may make the difference between success or failure in its implementation. There are misconceptions[20] concerning CASE, such as thinking of CASE as a methodology that replaces existing methodologies or techniques, thinking that

---

[19] Huff, Clifford C. "Elements of a Realistic CASE Tool Adoption Budget," Communications of the ACM. April 1992. pp. 45-54.

[20] Gibson, Michael L.; Snyder, Charles A., and R. Kelly Jr. "CASE: Clarifying Common Misconceptions," Journal of Systems Management. May 1989. pp. 12-19.

productivity gains are immediately evident, and expecting that CASE eliminates the data processing applications backlog.

An organization showing any misconceptions can move to acquire CASE for the wrong reasons. Benefits from CASE can only be achieved when a plan is carefully developed, realistic goals and expectations are set, and when an implementation strategy is followed.

Technology by itself does not produce miracles. No tool is useful if people do not want it, do not like it, or do not know how to use it. A good selection process of the right tool for the organization, hands-on support during the transition, and managing the learning curve of the new technology are vital to obtain expected benefits.

McClure[21] suggests a very comprehensive set of considerations to be covered within a CASE implementation plan, which includes management issues as well as technical issues. Also, she offers a set of guidelines for implementing CASE, beginning by establishing a definition for the software development life cycle and finishing by evaluating CASE impact. They are simply suggestions. An organization interested in CASE technology should review its objectives first, then define precise goals to achieve the objectives. In other words, the introduction of a CASE tool

---

[21] McClure, Carma. CASE is Software Automation. Englewood Cliffs, New Jersey: Prentice Hall, 1989. pp. 177-178.

must be part of the strategic planning process. Only then can a plan be developed to introduce this technology.

The acquisition of CASE technology is often triggered by productivity issues. Some organizations assume that the introduction of CASE will increase productivity. Indeed, it is considered a kind of granted benefit. But, in fact, it is not. Productivity can only be obtained as part of an integrated plan. Later, in Chapter III, productivity through the use of CASE tools will be covered in more detail. Right now, the idea is to state clearly the importance of acquiring a technology by knowing its potential. CASE can lead to productivity increases as a result of its application, not from an intrinsic characteristic.

Finally, organizations should pay attention to the learning curve of CASE. Learning how to use a new technology efficiently is never an overnight process; it takes time. In fact, the learning curve is a continuous process in the sense that people can come and go, different features are emphasized depending on the kind of project, others can be forgotten while others can be mastered. In any case, what is important to preserve and reinforce is the exchange of expertise between the users of the new technology.

Changes in the System Development Life Cycle

The Systems Development Life Cycle (SDLC) is a multi-phase process that normally consists of the following phases:

- Requirements Definition
- Analysis and Design
- Coding
- Testing and Implementation
- Operation and Maintenance

If these phases are properly followed, in theory, they lead to well-designed and easy to maintain systems. However, most of the time they are not conducted correctly because of time or budget constraints.

There is not a unique set of phases for the SDLC. But, two[22] characteristics are universal to all definitions for the SDLC:

1. In order to proceed to the following phase it is required to have the user approve of the current one, popularly known as the "sign-off" approval.
2. It is required to go according to a preset sequence of phases and within each, a preset sequence of tasks.

Usually, this traditional SDLC is known as the

---

[22] El Louadi, Mohammed; Pollalis, Yannis A., and James T.C. Teng. "Selecting a Systems Development Methodology," Information Resources Management Journal. Winter 1991. p. 14.

"waterfall model." According to this model[23], errors are found at every step of the cycle. Then, changes are made, new requirements might be introduced, and a lot of work is lost because of continuously going back to higher phases of the cycle.

Fisher[24] points out that there are time gaps and delays between phases in the SDLC. Sign-offs and personnel changes in the development team may create these gaps. This author also states that most failures in the SDLC are the result of poor planning, insufficient requirements analysis, and incorrect or incomplete design specifications. Also, not having or not following a development methodology is certainly a sign of potential failure.

Unrecorded system knowledge, insufficient analysis and early coding are causes of high-cost systems. Not only is cost a worry for IS managers, but also users' dissatisfaction, and the existence of incomplete systems. In the end, all this contributes to the backlog of systems.

Structured methodologies were developed to introduce structure and discipline to the development of systems. These methodologies emphasize the need for time for defining requirements carefully, and the devising of different design approaches before speeding up the coding. Formal

---

[23] QED Information Sciences, Inc. CASE: The Potential and The Pitfalls. Wellesley, Massachusetts. 1989. pp. 61-63.

[24] Fisher, Alan S. CASE: Using Software Development Tools. New York: John Wiley & Sons, Inc. , 1991. pp. 9-13.

methodologies are the backbone of CASE, providing a rigorous framework for correctly specifying and developing systems.

CASE helps in the entire process of developing systems or in any of its phases, from planning to implementation and maintenance. Through the use of CASE tools many analysis and design tasks are automated. Information about the system is stored to be used by people other than those who develop the system. Thus, systems knowledge is not lost.

Checking inconsistencies is also easier with CASE. This allows quicker correction before proceeding with coding. And even the time for this activity is reduced because of the automatic code generation capabilities of CASE tools.

To help understand the importance of CASE in the SDLC, McClure[25] presents a time comparison among the traditional software life cycle, with and without structured methodologies, and the software life cycle using CASE. This comparison follows.

---

[25] McClure, Carma. CASE is Software Automation. Englewood Cliffs, New Jersey: Prentice Hall. 1989. pp. 187-189.

TABLE 1: TIME CONSUMED PER ACTIVITY USING DIFFERENT APPROACHES FOR THE SDLC.

Activity	Traditional SDLC without structured methodologies	Traditional SDLC with structured methodologies	Using CASE
Analysis	20%	30%	45%
Design	15%	30%	40%
Code	20%	15%	
Test	45%	25%	15%

It is clear from the data presented in Table 1 that there is a considerable emphasis in the earlier phases of the SDLC using CASE, but, at the same time testing and coding can be done faster.

This author[26] also states differences between the traditional approach for systems development and that of software automation through CASE. CASE features are the reason for these changes. The traditional systems analysis methods, for example, have drastically changed. The way user requirements are defined changes using the prototyping capabilities of CASE, which allow the painting of screens

[26] Ibid. pp. 188-190.

and reports, and the quick construction of an executable model of the system. The users can use the prototype, and determine what they need and like and what they do not before actually having the system built.

CASE also helps to develop the systems specifications, analyze and check them, make the necessary corrections, and then continue to the coding phase. Elimination of errors during this process assures a decrease in the system testing time.

Maintenance activities are also affected. In the traditional SDLC, maintenance takes a lot of time and the main changes occur in coding. In the CASE SDLC, maintenance occurs at the design level, changing some systems specifications but having the facility of automatic code generation. This way time consumed in maintenance is reduced and no systems knowledge is lost.

Finally, under the CASE SDLC the concept of reusability emerges. CASE allows systems knowledge to be kept to perform subsequent enhancements of a system or the development of another. Project plans, data models and design specifications, for example, can be used again because they are stored and can be accessed by any software developer.

The new SDLC resulting from the incorporation of CASE tools is perfect for new systems to be developed. But, what

happens with the existing systems? . Bush[27] points out that many IS departments have more than 80% of their programming resources devoted to maintenance; therefore, CASE is also needed for existing systems to automate programmers tasks. This fact is also supported by Arturo Maria[28], who says that CASE has not been widely adopted for maintenance activities.

To this respect, experience shows that unless comprehensive systems documentation is kept, maintenance tasks are very difficult and costly to accomplish. Changes are required at the specification level, where programmers sometimes have to guess from existing code the system design ideas, and then proceed with the modifications.

The application of the reengineering concept can make the difference for those systems. What is achieved through the reengineering process is to take something from an existing system, such as planning specifications, analysis or design specifications, applications or user documentation, and bring it to the corresponding CASE tool (upper, middle or lower CASE). It can then be modified or improved. Then, it can be regenerated for easier maintenance in the future.

---

[27] Bush, Eric. "CASE for Existing Systems," InfoStrategy: The Executive's Journal. Spring 1991. p. 32.

[28] Maria, Arturo. "CASE Technology: Today's Reality," Journal of Systems Management. February 1991. p. 18.

As far as this is concerned, Gibson, Snyder, and Carr[29] state that usually reverse engineering in CASE and ICASE environments involves the change of

> "...analysis and design specifications and either translating those changes backward into planning specifications or translating changes in systems development specifications backward into analysis and design specifications and then into planning. "

Through automated reverse engineering using CASE less programmers are needed for maintenance. Thus, they can be assigned to new projects. CASE will help in facilitating the maintenance of those existing systems vital for an organization, if they are converted to this technology through reverse engineering.

Current Use of CASE

Vendors promote CASE as the ultimate solution to IS problems, including an increase in productivity. Experience has shown that acquiring the newest technology available does not always imply getting all the promised benefits. Not all the current CASE tools cover the full software life cycle and cannot link previously developed non-CASE systems into a CASE environment.

In a recent survey[30] of 430 Chief Information Officers

---

[29] Gibson, Michael L.; Snyder, Charles A., and Houston H. Carr. "Why CASE Belongs to Strategic Business Management," InfoStrategy: The Executive's Journal. Winter 1991. p. 18.

[30] Anthes, Gary H. "User Role Gains CIO Backing," Computerworld. February 24, 1992. p. 63.

(CIOs) conducted by Deloitte & Touche's Information Consulting Group, it was clear that the importance attributed to CASE has fallen sharply over the past two years. Instead the importance of user involvement has been stressed.

Evidence, however, shows that CASE tools are still being used for the sake of improving IS activities. Surveys have been conducted to assess the use of CASE and what benefits are actually reported.

One of these surveys[31] was conducted in 1989 in 63 organizations listed in the Directory of Top Computer Executives. The results show that 24% of the respondents were using some type of CASE tool. All the respondents using this technology thought that CASE had been beneficial. These results are backed by another survey[32] of 71 systems analysts conducted this year. In this case, survey results indicated that 66.2% of respondents were using CASE. The benefit most commonly cited was the efficiency in front-end application development. The survey showed that features such as project management, prototyping, documentation and graphics were the most commonly used by the respondents.

---

[31] Necco, Charles R.; Tsai, Nancy W., and Kreg W. Holgeson. "Current Usage of CASE Software," Journal of Systems Management. May 1989. pp. 6-11.

[32] "CASE Use Is Growing, but in Surprising Ways," Datamation. May 1, 1992. pp. 108-109.

In terms of productivity, Loh and Nelson[33] have found that gains vary depending on the suitability of a project and the acceptance of CASE. This was a result of a survey conducted by the University of Houston in 1989. In this case, CASE tools were applied mainly to the front-end of the SDLC among the companies surveyed.

A mail survey[34] of 400 CASE users was also conducted recently. Eighty-nine responses were received. According to the answers given by survey respondents, easier modification of preliminary designs, better standardization of resulting systems and easier documentation were pointed out as the main benefits obtained with the use of CASE.

Additional surveys, other than those mentioned above, have been conducted. What is important to conclude is that even with shortcomings, CASE tools actually help in the development of systems. Current tools are very likely to evolve and include more powerful features but, for now, an intelligent IS manager should take advantage of what is available in the market.

---

[33] Loh, Marcus, and Nelson, R. Ryan. "Reaping CASE Harvests," Datamation. July 1, 1989. pp. 31-34.

[34] Yellen, Richard E. "What Do Users Really Think about CASE?," Journal of Systems Management. February 1992. pp.16-17.

CHAPTER III

CASE AND PRODUCTIVITY

The last chapter reviewed data and research on how IS improves organization's operations. In fact, experience shows that by using better IS, people can do their job more efficiently. A valid question would be: "What about the productivity of those in charge of developing these systems? ".

Certainly something has been done for systems developers in order to increase their productivity. In a sense, the acquisition of new technology was regarded as a means for increasing productivity. CASE has been seen as the new miracle for improving IS productivity. Graham[35], for example, points out that CASE tools can not only increase productivity but can also improve the quality and longevity of the applications produced. On the other hand, Vessey, Jarvenpaa and Tractinsky[36] state that studies have

---

[35] Graham, Carol. "CASE Cracks Applications Backlog," Datamation. March 15, 1991. p. 17.

[36] Vessey, Iris; Jarvenpaa, Sirkka L., and Noam Tractinsky. "Evaluation of Vendor Products: CASE Tools as Methodology Companions," Communications of the ACM. April 1992. pp. 92.

found that while software developers believe CASE tools improve productivity, in fact they have no significant effect on productivity and have a relatively weak effect on specification quality. Perry[37] backs this by saying that CASE promises to improve productivity, but few organizations have realized significant productivity increases. According to this author, the initial phases of software automation using CASE improves quality; and the latter phases improve productivity, of the resulting systems.

It is evident, then, that a consensus has not been reached on whether CASE tools lead to productivity gains. Maybe one reason for this disagreement stems from the definition of productivity and the fact that the measures used are not always explained. Productivity definitions and examples of measures will be discussed in this chapter. Examples of success stories using CASE will be presented as illustrations of the potential benefits from the appropriate use of CASE tools.

---

[37] Perry, William E. "Make an Investment in Your ADP Workers," Government Computer News. October 14, 1991. p. 86.

Productivity Definitions

According to Scudder and Kucic[38], practical methods for measuring and improving IS productivity are still not readily available to managers, even though there is plenty of literature about it. Moreover, they also state that many organizations are facing difficulties measuring the performance of existing systems. For these authors, efficiency and effectiveness are key concepts for productivity. They define efficiency as being related to the resources consumed in producing a given application in a timely way. Effectiveness is defined as a concern with the quality of the product and its appropriateness to the situation for which it was designed. Following this rationale, they[39] state that IS performance can be:

- efficient but not effective
- effective but not efficient
- neither effective nor efficient
- both efficient and effective

Bouldin[40] defines productivity as the product produced divided by the resources used. She also states that in many successful CASE implementations, the company's commitment to

---

[38] Scudder, Richard A. and A. Ronald Kucic. "Productivity Measures for Information Systems," Information and Management. May 1991. pp. 343-344.

[39] Ibid. p. 344.

[40] Bouldin, Barbara M. "What Are You Measuring? Why Are You Measuring It? ," Software Magazine. August 1989. pp. 31-35.

quality was reported as the key element for success. To this respect, she adds that both quality and productivity should be obtained. Furthermore, Hubbard[41] says that, in the field, the question is whether higher productivity is the right goal; quality rather than productivity is the expectation for most middle and lower managers.

Therefore, before discussing the appropriateness of both concepts, productivity and quality, it is convenient to concentrate on each one. In order to do that, several definitions of productivity are presented, as well as a discussion of the quality factor[42].

Definition # 1.

> Productivity is a measure of how effectively the total assigned resources are used to produce desired products.

This is the classical definition for productivity. It is the same presented by Bouldin earlier. The desired products are IS, meaning data stored that will be used later (information) to accomplish a specific function within the organization. Here, productivity is a measure of the effectiveness of the information-generation resources in producing the right information desired for making decisions.

---

[41] Hubbard, Craig. "Increased Productivity Isn't Always Number One," Computing Canada. May 24, 1990. pp. 29, 32.

[42] QED Information Sciences, Inc. CASE: The Potential and The Pitfalls. Wellesley, Massachusetts. 1989. pp. 74-81.

Definition # 2.

Productivity is defined as:

$$\frac{\text{Output}}{\text{Input}} = \frac{\text{Value of the output products produced}}{\text{Assigned cost of producing output products}}$$

This is the classical definition of productivity in terms of costs: benefits divided by costs.

Definition # 3.

To increase productivity means to increase the yield of results, benefits, or profits, while following processing standards with sufficient quality control, to satisfy the specified requirements at a lower cost.

The problem here is that IS productivity is not like manufacturing productivity, where items are being produced less expensively as long as they are sold at the same cost. This is not what happens for IS products (information) because their value resides on their accuracy and timeliness.

Definition # 4.

Productivity = Efficiency X Effectiveness

where

Efficiency = Product / Cost

Effectiveness = Useful Products Delivered / Production

This definition can also be stated as useful results delivered divided by cost. The value of this definition is extended to include efficiency and effectiveness.

Efficiency is the unit cost of accomplishing the results, and effectiveness is the amount of useful results delivered per production unit. These definitions for efficiency and effectiveness are essentially the same given by Scudder and Kucic earlier in this chapter. Eventually, the problem of measuring productivity becomes the problem of measuring efficiency and effectiveness, which is explained in the following sections[43].

Measurement of efficiency.

An improvement in efficiency is obtained when the production of defined outputs is made with a lesser amount of human, software, and hardware resources. That is, efficiency is important since through its measurement it is possible to check if the systems are being developed at the lowest possible cost. General guidelines to measure efficiency are to be considered[44].

1. Analyze the systems information flow
   - identify critical inputs
   - identify system activities
   - identify critical outputs
   - develop indicators that are related to the work flow, and are reportable and repetitive
2. Establish a work product reporting systems
   - use computer system to generate counts

---

[43] Ibid.

[44] Ibid. p. 77.

- apply quality control criteria
- use regular production reporting to measure production

3. Allocate costs to production steps
   - identify cost elements
   - develop standard costs for functions
   - agree on cost allocation algorithms
   - collect cost data to measure costs
4. Apply quality standards and controls
   - measure objective quality (timeliness, accuracy, response, etc.)
   - measure perceived quality (user acceptance and complaints)
   - develop level of acceptable quality
5. Analyze efficiency
   - relate cost data to production data, where standards have been met
   - develop unit costs
   - compare against standards and over time

Measurement of effectiveness

Effectiveness is obtained when the outputs help in getting the organization's goals. Through its measurement it is possible to find out the fulfillment of the purpose of the desired results. Effectiveness is important because its measurement tells if the organization's resources are being applied to the most profitable and useful IS. The potential

problem in measuring effectiveness is its nature; it frequently depends on opinions. It is to the satisfaction of the users that functions are being performed in a more superior way.

In measuring effectiveness[45] the following guidelines can be used as examples.

1. Define the systems objectives
   - results to be delivered
   - quality level to be obtained
   - indicators of useful results delivered
2. Relate to efficiency measures
   - use same work/product reporting
   - get figures from regular production reporting
   - apply scale to quality standards
3. Collect data on user perception of effectiveness
   - apply scale to objective quality measures (timeliness, accuracy, response time, consistency, reruns, etc.)
   - conduct routine surveys of user perception of the accomplishment of the desired results
   - apply statistical scale to the data related to efficiency
4. Analyze the effectiveness and efficiency ratios
   - reject areas of strong dissatisfaction (non effective)

---

[45] Ibid. p. 80.

- compare the statistical effectiveness data to the measured efficiency data
- analyze whether results are homogeneous and normal, or if they consist of irregular, outlying data.

To complete the discussion of measuring productivity it is necessary to recall that quality rides in parallel with productivity and also that it must be another factor to measure.

Quality is even more difficult to define and measure. Quality could be interpreted as how valid or appropriate the system is, how correct was the system definition and development or how much control the system exerts in the activities it controls. Quality is a measure of the general usefulness of what is produced and the conditions under which this product was obtained. Factors such as accuracy, completeness, consistency, error tolerance, security, traceability, and auditability are key criteria for accepting the final version of a system.

Many organizations have developed what is called Quality Assurance (QA) programs. Usually QA is a staff function within IS that analyzes, develops, and implements control, and reviews IS. Some guidelines for QA follow[46].

1. Review appropriateness of all operations.

---

[46] Ibid.

2. Maintain adequate controls over computer use.
3. Set, maintain, and review proper quality control standards and procedures.
4. Determine that the information processed meets operational, decision, and technical needs, according to the specifications and requirements.
5. Have written, approved procedures and trained employees.
6. Be aware of users opinions and attempt to improve the user's perception of the operations.

Kucic and Scudder[47] do not make any differentiation between effectiveness and quality. But they do point out that effectiveness or quality is difficult to measure. Even more, they also state that according to the Quality Assurance Institute, less than half of 69 companies interested in quality measurement have a formal measurement program in place. According to these authors, the time spent in maintenance of existing systems is more than 45%. If quality is defined as low maintenance, it would be obvious to conclude that building higher quality systems is a must.

Regardless of whether effectiveness and quality are the same factor, something should be done to measure

---

[47] Scudder, Richard and A. Ronald Kucic. "Productivity Measures for Information Systems," Information and Management. May 1991. P. 344.

productivity. Smith[48] offers the following thoughts:

1. The excuses in large organization for not measuring productivity are mostly bureaucratic in nature, and contain very little substance.

2. Many people look for productivity as given by new technology. The first thing to do is to have a way to measure the current productivity level in the organization.

3. Less than 5% of all IS departments have any type of productivity measurement program.

4. If it is decided to use a CASE tool in a project to see if there is an increase in productivity, start by measuring the productivity achieved on a similar project in the past that did not use CASE.

5. No matter what approach to measurement is selected, it should be implemented consistently.

6. The primary objective is to measure productivity, thus change can be implemented and the organization remains competitive to survive.

To conclude this discussion, Bouldin[49] recalls the result of a famous productivity study, conducted during the 1950's, at a Western Electric plant in California. The conclusion of this study was that the act of measuring can improve productivity. This is known as the Hawthorne

---

[48] Smith, Al. "No Measure, No Change," Computerworld. October 8, 1990. p. 70.

[49] Bouldin, Barbara. "What Are You Measuring? Why Are You Measuring It? ," Software Magazine. August 1989. p. 37.

Effect. It is important to underline the fact that, most of all, an organization interested in measuring productivity should find the answers to the following questions: What do we want to measure? , What does productivity mean for us? , and What do we want to achieve in terms of productivity? .

Productivity Measures

Broadly speaking, Brathwaite[50] proposes that to compare the productivity of any two different technologies used in IS development, a measure of the system size to be developed must be defined. As he states, it would be ideal to have a measure independent of any technology or methodology. However, this is hardly the case, because each technology or methodology has inherent differences.

A very complete set of productivity measures is given by Kucic and Scudder[51]. A summary of these measures follows.

Systems performance measures.

These measures are useful to determine overall technical system operability, predicting the impact of proposed applications, and identifying hardware performance bottlenecks. Examples of such measures are response time,

---

[50] Brathwaite, Kenmore S. Applications Development Using CASE Tools. San Diego, California: Academic Press, Inc. p. 143.

[51] Scudder, Richard A. and A. Ronald Kucic. "Productivity Measures for Information Systems," Information and Management. May 1991. pp. 345-352.

workload volume capabilities, and network queuing algorithms.

Development measures for software.

Programmer measures are very familiar in the IS field, for example:

- Productivity: lines of code per staff hour
- Cost: staff hours per executable statement
- Reliability: errors per line of code, logical faults per line of code
- Maintainability: modules or units affected per change, staff hours to implement change.

Lines of code is probably the most known measure, however, it introduces some problems. For example, there is no universal definition for it. Moreover, applications developed using 4GL or CASE tools cannot be easily compared with applications using earlier languages.

Jones[52] gives more details concerning programmer productivity. He presents lines of code, cost per defect and ratios established for programming subactivities as the common metrics. All of them have problems. This author calls attention to the fact that a clear distinction between economic productivity and commonly known productivity does not exit. Economic productivity is defined as the amount of goods or services produced per unit of labor or expense. In

---

[52] Jones, Capers. "How Not to Measure Programming Productivity," Computerworld. January 13, 1986. pp. 65-76.

terms of programming, this kind of productivity is the functionability delivered to users per unit of labor or expense. Following this train of thought, a line of code is not an economic unit of measure. Commonly known productivity, on the other hand, is explained as finishing a task as rapidly as possible. To this respect, high-level languages actually improve coding speed.

Jones[53] continues stating that what is important is not how fast a program can be developed but how fast the program functions can be delivered. Most of the times, errors outside the code are more significant than those within the code. Moreover, when comparisons are made between projects developed using different languages, lines of code certainly is not a reliable measure. Even worse, when comparisons are made, sometimes there is no unique definition for lines of code. For example, for some IS professionals lines of code would include only executable lines, for others, they would include not only executable lines but also data definitions and comments.

Cost per defect is not a very reliable measure either. Cost per defect decreases when the number of defects increase. The result is that as program quality improves, the cost per defect will increase; consequently, quality will be penalized. Is this reasonable? .

Finally, the use of ratios or the assignment of

---

[53] Ibid p. 72.

percentages is also a common practice for measuring productivity. As Jones[54] states, for example, a modern rule of thumb for developing programs using high-level languages assigns 40% of the total time consumed for design, 35% for coding, and 25% for integration and testing. Unfortunately, percentages are not reliable since they can change drastically when code written in different languages is compared. Furthermore, when programs are novel or unique the staff will not have experience in dealing with them and existing ratios or percentages will not have any meaning.

Albrecht's function point analysis.

This technique is more independent of the technology used since it takes into account the external functions or project deliverables of an application. These are:

- number of inputs (forms, screens)
- number of outputs (reports, screens)
- number of inquiries a user can make
- number of logical data files used by the system
- number of interfaces to other applications

In order to develop a function point index, each deliverable is assigned a numerical complexity level. After that, factors influencing the project are assessed numerically. These factors are, for example, innovation, telecommunications, and distributed databases. Factors are added and expressed as a percentage. The analysis of the

[54] Ibid. p. 71.

percentages is as follows: those percentages less than 100% imply a positive influence, those greater than 100% will be interpreted as causing the software development project to take longer. Finally, the total influencing factor is multiplied by the total of the deliverable points index.

According to Bouldin[55], this measurement, invented by Al Albrecht from IBM, has obtained the reputation of measuring productivity in user terms. This method is the most widely accepted. Its advantage is that a manager can look at the installed base of systems as a dollar figure and determine its worth to the organization. But, she also adds that even when there is a number of developers satisfied with this metric, there are others -the majority- that are not using it as a metric. Instead, they are measuring productivity depending on the characteristics of the CASE tools they use. Some developers interviewed were using generators and what they do is to measure estimates against actual, states Bouldin[56].

Arthur's performance measures.

These measures target software quality. This method is similar, in a way, to function point analysis. Both measures require the existence of a database of historical software development data, and call for measuring software

---

[55] Bouldin, Barbara. "What Are You Measuring?, Why Are You Measuring It? ," Software Magazine. August 1989. pp. 31-32.

[56] Ibid. p. 32.

complexity. Arthur's method focuses on the internal software mechanics, while Albrecht's focus is software utility.

Arthur's method introduces two forms of measures. One refers to executable lines of code (ELOC) and the second measures complexity. ELOC has the same problems of lines of code, presented earlier. The basis for complexity measures is the commonly used programmed decisions such as IF-THEN, DO-WHILE, and GO-TO, and logical operators such as AND, OR, and NOT.

An organization using this method should develop a historical database of metrics such as mean time between failures and mean time to repair, for different types of coding structures. Hence, historical data could be used to pinpoint the types of systems likely to end up with the most problems. The most relevant idea here is to study the contents of the database to avoid making similar mistakes in the development of new systems.

Budgetary performance measures.

This type of measure has been commonly used in the IS field. The main point here is to relate project cost to benefits. If the benefits are greater than the costs, then the project is considered successful. Return On Investment or Return On Equity are used to measure organizational performance. This method also has limitations. What is measured here is the productivity of capital, and this does

not tell a lot about how IS are helping the organization.

Another way of measuring projects is in terms of time and budget. To this respect, a certain amount of budget and an established time to be completed are assigned to software development projects. Comparisons between time and budget planned against their actual counterparts determines its success.

After discussing the different measures available, it becomes evident that none is reliable enough to be used by itself. This is the reason why multiple measures should be used if an organization wants to find out its current level of productivity.

Kucic and Scudder[57] continue their discussion of measures presenting the Capers Jones' set of IS organizational performance measures. This set is broader than the previous ones. His set comprises:

1. Defect removal ratio.
2. Maintenance productivity.
3. Successful product ratio.
4. User satisfaction.
5. Employee satisfaction.
6. Staff training.
7. Development productivity.

Some of these measures need further explanation. For example, defect removal ratio is calculated by comparing the

---

[57] Scudder, Richard A. and A. Ronald Kucic. p. 348.

number of defects found by users to the number of defects found by the developers prior to release the system. The successful product ratio is defined as the number of projects completed (and used) as a percentage of the number attempted. Employee satisfaction and user satisfaction can be estimated through surveys.

The authors also refer to the Dickson and Wetherbe's set of measures for IS performance. This set introduces two new elements. One is the need to set a performance goal for measuring planning for systems resources capacity and the other is the focus on managerial performance. A summary of this set follows.

1. Financial performance.
   - budget performance
   - cost recovery
   - distribution of costs by industry standards
2. Organizational efficiency.
   a. Development performance
      - meeting project time and cost goals
      - staff turnover
      - size of system request backlog
      - system maintenance cost
   b. Operational performance
      - system availability/downtime
      - late jobs
      - on-line response time

- system utilization
- throughput
- job reruns

3. Managerial performance.
   - attitudes of senior management
   - attitudes of user managers
   - performance evaluation by external assessors
4. Other.
   - availability of capacity in systems resources to meet future operational and development requirements

Through the study of the previous set of measures and the experience of several organizations, Kucic and Scudder have developed the following comprehensive set of measures.

1. Personnel performance.
   - technical capabilities
   - business knowledge
   - training
   - replacement projections
   - career satisfaction
   - IS job satisfaction
2. Managerial performance.
   - attitude of senior management
   - attitudes of users
   - performance audits
   - perceptions of IS "problems"

- perceptions of IS "capabilities"

3. Developmental performance.

   a. Quantitative

      - time and cost
      - staff turnover
      - size of system request backlog
      - system maintenance costs
      - system cost standards
      - lines of code/executable lines of code
      - feature point analysis
      - function point analysis
      - charge out performance

   b. Qualitative

      - application portfolio
      - formal methodology quality
        - SDLC
        - structured design
        - project control
        - productivity aides
        - documentation quality
      - team size
      - user interaction

4. Goal setting.

   - senior management role in IS planning
   - IS representation in planning
   - quality of planning

- balance of risk - portfolio management
- forecast for future technology
- forecast of future IS capabilities

5. Operational performance.

   a. Quantitative

      - system availability
      - late jobs
      - job rerun percentage
      - throughput
      - system utilization
      - maintenance performance ratios

   b. Qualitative

      - backup performance
      - security
      - privacy
      - user interaction
      - complete data
      - accurate data
      - understandable output
      - timely output
      - relevant output
      - user friendly operations
      - error resistant operations

6. Financial performance.

   - budget performance
   - cost recovery

- distribution of costs
- market-based industry standard costs
- expense categorization
- IS investment model

Kucic and Scudder propose the above set of measures as a starting point. For an organization to select and implement those measures, it must have its own meaning of productivity. That is, an organization should determine first what is worth to measure, select measures accordingly, and implement a consistent program for continuous measuring and feedback.

As an example, the authors present the case of a company that wanted to implement some measurement program. In order to do that, IS activities were assessed, and interviews were also conducted to determine the appropriate set of measures for this organization. The following set of corporate IS performance measures was suggested.

1. System availability.
   - percentage of system availability
   - uptime/downtime ratio, throughput, jobs processed
2. Error rate.
   - ratio showing the percentage of errors found by users versus the percentage of error found by IS prior to release
3. Development effectiveness.
   - function point per staff member per month

- comparison to national standards
- documentation effectiveness

4. Maintenance effectiveness.
   - function point per staff member per month
   - comparison to national standards
   - repair of error prone modules
   - reduction in the number of job reruns required
5. Staff satisfaction.
   - job satisfaction survey
   - training level
   - turnover ratio
6. User satisfaction.
   - system response time
   - response for equipment requests
   - communication with IS staff
   - user satisfaction survey
7. Budgetary performance.
   - corporate wide performance to budget
   - comparison to industry standards
   - market-based charge-out
8. Data/information availability.
   - connectivity levels
   - information response speed

As a conclusion, the authors advise that IS performance measures be available to top management in a way that they can be understood from a business perspective. Graphics are

a very effective way of presenting results, however different formats should be tested until the appropriate ones are identified.

The objective in presenting the above example is to illustrate how an organization should proceed when establishing productivity measurements for IS activities. Measures, of course, should be adjusted depending on the characteristics of the technology used. CASE, for instance, has its own characteristics and potentiality that must be assessed in a comprehensive measure. This measure should target the effects in terms of productivity that this technology would bring to the IS environment.

Mosley[58] offers in his article a relevant comment regarding productivity. He emphasizes that productivity gains using CASE cannot be measured. He points out that there is an abundance of literature on "success stories". But when probed about how this success was measured, it was found that success was based on estimates of productivity.

According to him, measurement of CASE productivity is different because there is a lack of historical data to compare it to. Moreover, most organizations do not keep this kind of data because of the pressure to develop new systems. He advises that in order to demonstrate CASE success, an organization should measure before and after

---

[58] Mosley, Daniel J. "Getting CASE Straight: Can You Use It or Not? ," Chief Information Officer Journal. Fall 1990. pp. 58-59.

CASE. His suggestion is that an organization should:

1. Decide which parameters form the baseline and begin to recognize them immediately. Also, a small but critical set of development characteristics should be identified.
2. Discard parameters that prove not to be useful and replace them with others.

A question is unavoidable. After the introduction of CASE technology, when would an organization begin to perceive gains in productivity? . Gibson, Snyder and Kelly[59] indicate that "because much time and effort will be initially spent on performing start-up clerical work, tremendous productivity gains will not be evidenced until CASE is firmly established. " In fact, according to them, productivity gains will increase the longer an organization uses CASE to build systems.

CASE Implementation Stories

Literature about CASE is abundant. There is evidence of success stories in implementing CASE as well as failed attempts. The idea of this section is to provide some case studies of organizations that have used CASE tools and whether or not they have experienced some improvements in the software development process.

---

[59] Gibson, Michael L,; Snyder, Charles A., and R. Kelly Jr. "CASE: Clarifying Common Misconceptions, " Journal of Systems Management. May 1989. p. 16.

The following examples of increased software productivity are taken from McClure's book[60].

Excelerator productivity experiences.

Excelerator, from Index Technology Corporation, is an analysis toolkit. It offers an integrated set of tools for automating analysis and design tasks. Excelerator has the following facilities:

- an automated diagramming tool for drawing structured diagrams
- screen and report painters for specifying and prototyping the user interface of a system
- an integrated repository for storing and cross-referencing all analysis and design information
- an automated analysis tool for checking and reporting the syntactic correctness, completeness, and consistency of structured diagrams.

The results, according to McClure, of recent studies of 12 organizations using this tool indicate productivity increases ranging from a factor of 2 to a factor of 10 times. The development environment of these 12 organizations was similar (IBM mainframe, DBMS such as IMS and DB2, COBOL languages for large applications and use of 4GL such as Focus). These organizations were using the tool mainly for automating currently used structured methods.

---

[60] McClure, Carma. CASE Is Software Automation. Englewood Cliffs, New Jersey: Prentice Hall. 1990. pp. 158-168.

The primary user was a system analyst and the main use was for system requirements specification. Users of this tool reported needing an average of 8 to 30 hours to be proficient using this tool; they were already knowledgeable in structured methodologies.

Excelerator users were convinced that the key elements to productivity gains are good management, solid understanding of structured methods, and powerful automated tools.

1. U. S. Government Organization.

This was one of the 12 case studies. It reported a great reduction in the length of the SDLC. Systems specifications that were expected to be completed in two years were done in four months.

2. Touche-Ross.

Touche-Ross of Milwaukee, Wisconsin, reported dramatic changes in the way they do analysis tasks. There was a feeling that the major productivity advantage of using this tool is the facility of reviewing systems requirements. Another advantage cited is the consistency of the output produced by the tool.

Application Factory productivity experiences.

Application Factory from Cortex Corporation is used to develop on-line, multi-user IS. It runs on DEC VAX computers. Its core is an automatic code generator. The facilities offered by Application Factory are:

- screen and report painters for specifying and prototyping the system's user interface
- a repository for storing all information about a system
- automated checking for completeness and consistency of program specifications
- a code generator capable of automatic generation of 95% of the program code from program specifications
- an automated program documentation generator

This tool is used by analysts and programmers to develop and maintain medium-sized-to-large systems. It is a very appropriate tool for these systems in high-volume transaction environments interfacing to real-time devices.

Case studies using this tool indicate an average productivity increase of a factor of 13 times with Application Factory over COBOL. These case studies included the building of 26 systems at 22 separate locations. Productivity gains were measured with function points by comparing the estimated Cobol effort to the actual Factory effort. System size ranged from 78 to 2418 function points. This is approximately the size range of 8300 to 253,000-line COBOL programs. According to these studies, the productivity relative to COBOL increases as application size and complexity increases. Project size and developer experience are elements affecting productivity.

DuPont.

DuPont Textile Fibers Plant of Wilmington, Delaware, has its own methodology for systems development. It is called RIPP - Rapid Interactive Production Prototyping. This methodology limits systems development projects to a maximum of 90 days. Sixty of those 90 days are spent in prototyping. The DuPont productivity goal is to improve productivity by a factor of 10 over the entire life cycle. Application Factory is helping them in achieving this goal.

DuPont reported great cost savings when they used the Application Factory for developing systems. In nine development projects DuPont saved almost $2 million using Factory instead of third-generation languages such as COBOL or FORTRAN.

According to DuPont, by using Application Factory, systems were built faster even when less experienced developers were in the project. The learning curve for the Application Factory was three months to reach competency, and six months to reach an expert level.

Information Engineering Workbench productivity experiences.

Information Engineering Workbench (IEW), from KnowledgeWare, has the following facilities:

- an automated diagramming tool for drawing structured diagrams
- a CASE repository with its intelligent information management system, called an encyclopedia, for

storing, analyzing, and coordinating all system information

- an automated analysis tool called the knowledge coordinator, for checking the completeness and consistency of all systems analysis and design information

IEW is a CASE workbench and methodology companion to automate a structured methodology called information engineering. In the case studies, this tool was mainly used to give automated support for systems analysis and design tasks of the information engineering methodology.

Deere & Co.

In this company, located in Moline, Illinois, there are 39 decentralized systems development groups and no overall company standards. Deere & Co. has developed its own methodology for building systems, which is based on the information engineering methodology.

IEW was chosen to help in achieving the company's productivity goal, which is to achieve a 30% productivity increase in the software development. The main user of the IEW is the system analyst, and the main use of the tool is to design new systems. An average of two days is reported as needed to become proficient with the tool, assuming that the developer already knows information engineering techniques.

Productivity increases of a factor of 2 for

requirements analysis and data analysis tasks using IEW rather than doing it manually have been reported. The company also reported that the most important factor in these gains was the automated checking for design errors in the structured diagrams. The advice from this company is to have the development methodology in place first, tools should come later.

The examples presented above suggest some important factors for achieving success when using CASE. For example, it is crucial to select CASE tools that are going to improve productivity where it is most needed. Productivity gains are the highest when the chosen tool automates a structured methodology already known and in use in the organization.

Secondly, training in the methodology as well as training for the use of the tool is vital. To this respect, methodology training seems to be more important than tool training.

Finally, good tools in the hands of experienced, and skilled developers are the key factors for success.

Bouldin[61] reports another success story taking place at Midland International Trade Services in New York. This company acquired MicroStep, a product of Syscorp International. MicroStep is a PC-based CASE tool that generates code from diagrams and specifications. The

---

[61] Bouldin, Barbara. "What Are You Measuring?, Why Are You Measuring It? ,"Software Magazine. August 1989. p. 32.

company obtained gains in productivity being able to develop a system five times as fast with this tool as with a traditional method.

Another example of success is presented by Moran[62]. Aetna Life and Casualty Co. at Hartford, Connecticut acquired Information Engineering Facility (IEF) from Texas Instruments. This tool is a workbench that can be used through the entire SDLC. This tool was used to roll out a database for its health plan business unit. According to them, the project took 13 months to finish. In comparison, an equivalent project using another technology would have taken two to three years and twice as many people.

On the other hand, Moran[63] also calls attention to the fact that CASE has been marketed as a silver bullet to solve every kind of problem in the IS environment. He reports that what was said by Thomas Pettibone, former CIO at New York Insurance Co., is a reality in many companies where CASE has been installed. Pettibone related that the first attempts made in the company with CASE resulted in failure. This damaged the credibility in the IS department. The main mistake was to believe blindly in CASE, and put it to work immediately in systems that were too big or too visible for it to handle.

---

[62] Moran, Robert. "The Case Against CASE," InformationWeek. February 17, 1992. p. 32.

[63] Ibid. p.29.

Evidence indicative of this fact is given as the result of a CSC/Index survey of 444 IS executives conducted in late 1990. According to this survey, 76% of the IS executives considered CASE as one of the most promising new technologies, but a year later, this percentage dropped to 19%.

Hayley and Lyman[64] indicate something similar resulting from a survey conducted in 1990 by Deloitte and Touche. Five-hundred and sixty-eight CIOs were asked to assess the use and the impact of CASE. The respondents tended to agree on the fact that CASE tools do not necessarily lead to productivity. CIOs rated higher quality systems as the most likely benefit they could obtain from CASE.

From the previous discussion it is obvious that CASE can help an organization or not. It is not the tool by itself that is going to bring productivity gains; the tool is only one element out of many to obtain it.

The next chapters describe how specific organizations are dealing with CASE regarding productivity. To this respect, Chapter IV describes the purpose and methodology of a survey conducted in several organizations where this technology is being used. The results of this survey are presented in Chapter V.

---

[64] Hayley, Katheryn J., and Lyman, H. Thaime. "The Realities of CASE," Journal of Information Systems Management. Summer 1990. pp. 18-19.

CHAPTER IV

SURVEY PURPOSE AND METHODOLOGY

Purpose

Most of the productivity measures have been defined and implemented for programming. As explained in Chapter III, lines of code and cost per defect are two of them. Lines of code, for instance, does not have a universal definition; depending on the definition used, productivity will vary for the same program. It is evident, then, that there is something more important than measuring the extent to which CASE increases IS productivity. Some questions arise: Are organizations measuring productivity in any way? Did anybody think how to measure productivity prior to the introduction of CASE tools in the organization? Was its introduction considered a part of the IS strategic plan? Overall, how do organizations define productivity?

The answers to these questions are important for everyone involved in the IS field. Therefore, the purpose of the survey presented here is to seek answers to the previous questions through the study of specific organizations where CASE tools have been used.

In order to conduct this survey, the following set of

hypotheses were established:

1. CASE was not introduced as part of the IS strategic plan.
2. Before introducing CASE there were no measures for productivity. Companies did not even have a clear definition of productivity.
3. Productivity measures were developed mainly for programming.
4. CASE tools by themselves do not lead to increases in productivity.
5. The SDLC has changed because of CASE tools' use.

The methodology followed is described in the following section.

Methodology

In order to accomplish this survey's purpose the following activities were required:

a. Literature review
b. Design of a questionnaire
c. Selection of organizations to include in the survey
d. Data collection
e. Analysis and interpretation of results

The result of the first activity was presented in chapters II and III and was the basis for the following activities. The next steps were the design of a

questionnaire, and the identification of organizations that would participate in the survey.

A questionnaire was designed according to the hypotheses that I had established and the literature review was conducted to identify main concerns and achievements using CASE technology. (See Appendix).

The questionnaire was divided into two main sections. The first one, Organization Data, was intended to gather general data of the organization and its information technology infrastructure (employees, software, hardware, IS architecture, etc.). The second section, Information Systems Activities, was targeted to collect data of IS personnel (number, years of experience, exposure to CASE and structured methodologies and their learning curve using CASE technology); IS development (methodologies used, type of projects, standards, statistics and the use of consulting companies); productivity issues (definition and measures for productivity) and the introduction of CASE tools to the organization and to the development of systems. Overall, this questionnaire targeted questions concerning the impact of CASE tools; when and how they were fully integrated in the IS development process, with special emphasis on productivity measurement.

The organization selection process was subjective and was not designed to provide a basis for statistical conclusions. A network was established to contact people in

organizations who could respond to the questionnaire. These organizations are representative of different economic sectors. Certain criteria should be met for an organization to be part of the survey:

1. It should have, at least, two years of experience using CASE technology.
2. It should be based in the Washington, DC area
3. It should be of what is generally considered as medium to large size.

Seven organizations that met these requirements were willing to take part in the survey, representing different interests such as international development, economic development, health, telecommunications, government agency, and IS consulting.

A number of IS professionals were selected to answer the questionnaire. They should have experience in at least one CASE tool, and be systems analysts, programmers or unit/department chiefs.

The questionnaire was given to 18 professionals, in 7 organizations. In most cases, a brief meeting to explain the purpose of the survey and to present the questionnaire took place at the interviewee's office. Once completed, most questionnaires were returned through ordinary mail.

In total, fifteen questionnaires were received. Then, a selection process was followed. Three organizations were selected as objects of further study. Their answers, their

quality, the sharing of similar characteristics, and the willingness to participate in an in-depth study were the criteria for this selection. The chosen organizations share two main characteristics. One, is that they have a diverse working force; that is, people working for them come from different countries. The other one is the nature of their activities, namely, international. They have a lot of interaction with foreign governments. The ultimate purpose of this selection process was to have a more balanced survey population, thus comparisons within this group would be analyzed on a fair basis.

Because of the small number of organization selected, the survey became an in-depth case study of these organizations. Additional interviews were conducted to clarify answers to the questionnaire when appropriate. Not only respondents were interviewed but also systems auditors and unit chiefs. The number of professionals interviewed from the organization depended on the organization's infrastructure, the IS activities and the use of different CASE tools. Further analysis of the results was the next step, which is presented in the following chapter.

CHAPTER V

CASE STUDY RESULTS

After the questionnaire collection, as it was explained in the previous chapter, a selection process took place. The purpose of this process was to determine what organizations would be given further study.

Three organizations were chosen. Their name will not be disclosed, confidentiality was requested. A brief description of each organization is presented in order to understand the kind of IS activities taking place there. The purpose of this chapter is to summarize the results of previously established hypotheses, with the information gathered during the study.

Organization Characteristics.

In this section the organizations activities, hardware and software resources, IS architecture, and IS development methodology are described.

Organization A.

This is a multinational organization. Its main purpose is to coordinate regional efforts to improve physical and mental health and to maintain close relations with national

health organizations in Latin America. It has a staff of 1600. Its headquarters are in Washington, DC.

Hardware resources: one IBM mainframe, two minicomputers, and approximately 200 IBM PC's.

Software resources: NATURAL, Cobol, and PL/1 as programming languages. ADABAS as the data base management system (DBMS). Several software packages such as Lotus 1-2-3, Quattro and Harvard Graphics. Excelerator, from Index Technology Inc., is the standard CASE tool available since 1986.

IS architecture: Decentralized since 1990. There is a centralized group for the development of administrative systems, and another group to maintain existing systems and to develop technical systems. Some specific applications are developed by consulting companies on a contractual basis.

IS development methodology: Yourdon structured methodology. Prototyping using NATURAL and ADABAS is widely used.

Organization B.

This organization is a multinational belonging to the economic development sector. Its main activity is to help accelerate the economic and social development of Latin-american countries. It sponsors projects that expand agricultural production, finance energy projects, develop industry, urban renewal, and health and education, and improve development institutions. This organization also has a staff of 1600. Its headquarters are in Washington, DC.

Hardware resources: Two mainframes, four minicomputers and more than 1500 IBM PC's. SUN workstations are also available.

Software resources: Programming languages such as Cobol, Fortran, PL/1, C, CLIPPER. DB2 as the DBMS. Some software packages run on PC's such as Lotus 1-2-3, for instance. Information Engineering Facility (IEF) from Texas Instruments is the standard CASE tool since 1988.

IS architecture: Centralized. Consulting companies are hired to develop specific systems. Some of them are developed using IEF. There is development of specific systems in some IS units established in the main departments, however, regarding CASE the

IS department is the principal user.

IS development methodology: Martin's information engineering.

Organization C.

It is a multinational established by the United Nations to assist in improving the standards of living in developing countries by facilitating financial resources from developed countries. It gives loans to governments or to private enterprises, with the guarantee of their governments, where private capital is not available on reasonable terms to finance productive investments. Its programs concentrate on rural and urban development, agriculture, and education. This organization has a staff of approximately 7000. Its headquarters are in Washington, DC.

Hardware resources: This organization has several IS working environments. As a whole, IBM and UNISYS mainframes, DEC VAX minicomputers, and IBM's (approx. 6000) and Macintosh (approx. 2000) personal computers are available. SUN workstations are also used.

Software resources: These resources are according to different hardware bases. For example DBMS such as DB2 (IBM), SQL/DS (IBM), DMSII (UNISYS), ORACLE (DEC), RDB (DEC). Programming languages such as Cobol,

Algol, C, PL/1, Pascal, and Fortran. CLIPPER and FoxBase are also available for PC's. Other PC-software packages used are: Excel, Lotus 1-2-3, Word, Wordperfect, MacProject, MacDraw and Harvard Graphics. The current standard CASE tool is Information Engineering Workbench (IEW), from KnowledgeWare. Its introduction date varied for different IS units. For some it was introduced in 1988, while others obtained it in 1989. DesignAid from Nastec Corp. (1986-88)and Excelerator from Index Technology (1987) were used in some units before introducing IEW as the standard CASE tool.

IS architecture: Completely decentralized since 1986. IS units across the organization work in a fully independent way. These IS units were organized according to the main business areas of the organization. There is no communication among them in the sense that a unit does not have any information of the systems being developed in the others. This fact is understandable if one considers the

different IS working environments there. It could be said that regarding IS operations, this organization behaves as many smaller organizations sharing the same buildings. There is, however, a corporate unit in charge of checking how IS activities are carried out in the IS units.

Some IS units hire consulting firms to develop specific systems.

IS development methodology: Martin's information engineering, object-oriented, and Gane/Sarson structured methodology.

A summary of the answers to the questionnaires is presented from Table 2 to Table 5. From these tables it is evident that the number of professionals interviewed is not the same in each organization. As it was stated in the previous chapter, IS infrastructure and the use of different CASE tools were factors to determine the number of professionals interviewed. Entry A shows the data provided by an IS professional working in the unit in charge of systems maintenance. The opinions expressed by this professional are extremely valuable since he has been working for a long time for this organization in an important position. He has witnessed the changes in the IS

activities and was the person who actually recommended the acquisition of a CASE tool.

The answers of two experienced professionals of organization B, who are in charge of important development projects, are presented in each table instead of a single entry. Their answers are also very valuable because of their knowledge of the organization and of the CASE tool used there. Both professionals work at the IS department. Entry B1 corresponds to the answers of the project leader in charge of a very important corporate system. The answers refer to the group this person manages. Entry B2 represents the answers of another project leader who has significant experience in this organization. This person sometimes substitutes for the IS chief. The answers of this person refer to the IS department as a whole. As far as this organization is concerned, it should be mentioned that analysts and programmers contracted on a temporal basis, are not considered as permanent members of the organization.

Organization C has six entries in each table because of the completely decentralized nature of its IS activities. Each entry represents the response from key professionals in those units where the IS development activities were stronger. Data presented on entry C6 refer to the IS activities on the organization as a whole.

TABLE 2: SURVEY RESULTS
Information Systems Personnel

	ORGANIZATIONS								
	A	B1	B2	C1	C2	C3	C4	C5	C6
Number Analysts/Programmers	6/3	3/3	19	7/1	7/8	2/5	3/4	50	600
Average years of experience	10/3	6*	15	9/3	9/6	9/5.5	6/5	8	6.5
Average years of experience in structured methodologies	2*	4.5*	3	9*	?	4/3	3/1	3	10
Number trained in the current CASE tool	3/1	3/3	14	4*	5/2	2*	2*	50	200
Learning-curve-length	?								?
Weeks									
Months			?*	3*	3*	1*	2*		
Years		1						2	

Notes:
? : Not Known
* : Only systems analysts
a/b : "a" refers to systems analysts
"b" refers to programmers

Data in column B2 refers to the IS department as a whole
Data in column C6 refers to organization as a whole

TABLE 3: SURVEY RESULTS
Information systems development

	ORGANIZATIONS								
	A	B1	B2	C1	C2	C3	C4	C5	C6
In-house development methodology for building systems				X			X		X
Applications developed:									
Types: Basic	X	X	X	X	X	X	X	X	X
Technical	X					X			X
Other					X			X	
Processing: Batch	X	X	X		X	X			X
Interactive/Real-time	X	X	X	X	X	X	X	X	X
Standards for type of application			X						
Statistics on applications development costs/time			X						
Statistics on maintenance			X	X					

Note:
The absence of a checkmark in a cell represents a negative answer.

TABLE 4: SURVEY RESULTS
CASE tools introduction

	ORGANIZATIONS								
	A	B1	B2	C1	C2	C3	C4	C5	C6
Approximate date for introduction of CASE tools	1986	1988	Aug. 87	Mid-87	March 88	July 87	1989	1986	1988
Number of CASE tools used through the years	1	1	1	2	1	2	1	2	2
Introduction was part of the information systems strategic plan		X	X	X	X	X	X	?	
Consulting company used for assistance in implementing CASE tools		X	X	X	X	X	X	X	
Plan developed and followed to introduce the use of CASE tools		X	X	X		X	X		
CASE tools fully integrated to the information systems development process	Not yet	1990-1	June 88	End of 87	?	Jul-Aug 88	?	Never	Not yet

Notes:
? : Not Known

The absence of a checkmark in a cell represents a negative answer

TABLE 5: SURVEY RESULTS
Productivity issues

	ORGANIZATIONS								
	A	B1	B2	C1	C2	C3	C4	C5	C6
Definition for productivity									
Measurements for productivity						X			
Changes in the information systems backlog after introduction of CASE						?			
Consulting companies used for systems development:									
using CASE tools	X	X	X	X				X	
not using CASE tools		X	X					X	

Notes:
? : Not Known

The absence of a checkmark in a cell represents a negative answer.

Testing of Hypotheses

The validity of the hypothesis was tested against the questionnaires' answers as well as according to the interviews conducted at each organization with survey respondents and other IS professionals.

Hypothesis #1: CASE was not introduced as part of the IS strategic plan.

This hypothesis was rejected. In two of the three organizations, the decision of acquiring a CASE tool was a strategic decision.

In one of these two organizations, a committee decided the introduction of the standard CASE tool for the sole purpose of building quality systems. The feeling there is that a good system will require less maintenance, and that it will take longer for changes to be requested after its release. Thus, in the long-run good systems become less expensive. Training was given to master the tool. During training, more emphasis was made in the concepts lying behind the tool than to the mechanics of the tool itself.

Also, regarding this organization, it is important to notice that another CASE tool was being used in some IS units before a standard tool was adopted. However, that previous tool was introduced indirectly, not as result of a corporate decision. Some consultants were working for this

organization, who already had experience with the tool. Thus, they suggested to buy it to improve the analysis and design phases of the applications being developed. The IS units had the resources to acquire the tool, and the vendor was responsible for the training. This tool was not widely used for systems development. Its introduction was not based on a specific plan, it just happened to help in specific systems. A short time after that, the organization decided to acquire a CASE tool that would be used as the standard for all IS units. This was a corporate decision. It had no relation to the experience of IS units that bought the CASE tool mentioned before.

Regarding the second organization, the decision of acquiring CASE was the result of a recommendation made by a consulting company. This company was studying the organization as a whole looking for ways to improve its performance. Top management decided to acquire this technology. A committee was formed to evaluate several tools and to decide which one would serve the IS activities better. A consulting company was hired to assist in the introduction of the tool. Training was given to technicians and to users. A pilot project was developed using the new tool before assigning it to the development of new systems.

The third organization acquired CASE indirectly. A consulting firm was hired to develop a system. For that system, this consulting company was using a CASE tool but

its use was not being reported to the organization. When this fact was discovered, an IS executive proposed the acquisition of a CASE tool. Two products were evaluated, one of them being the same tool used by the consulting company. That tool was the one selected for the organization. Training was given to use the tool. When the consulting firm finished the system, they gave a soft-copy of the system specifications to the organization. This allowed the organization to maintain the system at the specification level. At that time, the organization already had the CASE tool in place, which was the same used by the consulting company.

Hypothesis #2: Before introducing CASE there were no measures for productivity. Companies did not even have a clear definition of productivity.

This hypothesis was validated. None of the organizations has or have had a definition for productivity.

One of the IS professionals in organization C said some measures are used in her IS unit, but no definition for productivity exists. These measures are timely completion as scheduled, and sign-off from users. They are used in analysis, design and programming phases of the SDLC. All other interviewees in this organization and in the other two said they do not have any productivity measurements. When asked to be more specific, some pointed out that "best effort" is the implicit measure used. Others stated that

comparisons between scheduled time and actual time and comparisons between planned budget and real costs have been in some cases, the only measures used.

One of the IS professionals interviewed said that there has not been time for measuring the success of projects developed using CASE. Users still have the feeling that everything goes slowly. But, at the same time, users are asking for more systems every day.

Another interviewee from another organization complained about top management. According to this person, management wants "quick, cheap and good systems, " but they do not understand how difficult it is to obtain these three factors at the same time. They do not understand the importance of quality as part of productivity. Despite this situation, IS management is more concerned with the third factor, that is, good systems.

Hypothesis #3: Productivity measures were developed mainly for programming.

Neither was this hypothesis rejected, nor could it be validated. As explained earlier, none of the organizations has formal measurements for productivity. According to the survey, only one IS unit in organization C has some measures but they are applied equally to analysis, design and programming phases. Therefore, there is not enough evidence to test this hypothesis properly.

Hypothesis #4: CASE tools by themselves do not lead to

increase in productivity.

What happens with this hypothesis is similar to what happens with hypothesis #3. That is, if there is no definition for productivity and no measures for it, it is not possible to reject or validate this hypothesis. Moreover, there is no historical data at hand to compare if IS activities are being performed more efficiently now. For example, regarding maintenance, one of the interviewees said that they have a file of maintenance requests for systems but there are no statistics available for this. If some statistics are desired, one should search the file looking for systems requests forms and deduce the statistics from there. This could become a very time-consuming task. The obvious result is that nobody has time to do it because of the continuous requests for more maintenance and new systems.

In one of the two cases where statistics on applications development such as costs and time were kept, the interviewee said that the nature of the systems and specific development circumstances do not allow a meaningful comparison.

The other case was more interesting. Since mid-1988 statistics were kept. In this IS unit what is understood by maintenance is how many "bugs" are reported in systems already in use. Requested system improvements are understood as enhancements, not maintenance. Statistics on

both are kept, and reported every three months to management. These reports also show what systems are on the "waiting list" to receive maintenance or to be enhanced. Neither cost nor time consumed to perform these activities is shown in the reports. Even when this information is kept, it is not used as feedback to evaluate the effectiveness of the IS development process. However, the interviewee agreed that because of the methodology behind the CASE tool used, systems developed using it are very stable and easy to maintain.

Information obtained through the questionnaires and interviews suggests that there is not a noticeable change in the IS backlog after the introduction of CASE tools. However, one of the interviewees believed that, users are now asking even for more systems knowing the potential of CASE tools to facilitate the work of IS professionals.

It is important to notice that all the organizations hire companies to develop systems. It could be deduced then, that IS departments cannot meet all organization's information needs. But, this is not always the case. Some systems are so singular and are needed in such a short time that an experienced company is hired to actually built them.

Hypothesis #5: The SDLC has changed because of CASE tools.

This hypothesis was validated to some extent. Because of CASE tool characteristics some changes have occurred in

the way systems are being developed. For example, IEW and IEF are based on Martin's information engineering methodology. Therefore, the approach to develop a system changes if this methodology was not the one used in the organization prior to the CASE tool introduction. Prototyping capabilities of CASE have also influenced the changes. The user evaluates the system before IS professionals develop it. This helps in having users more satisfied after the system is released.

Regarding Excelerator experience, the organization using it reported that it is used more like a graphics tool than in any other way. Yourdon's structured methodology is widely used for developing systems and Excelerator supports this methodology. Consequently, the tool has not changed dramatically the way systems are developed. More than the CASE tool, the combination of ADABAS and NATURAL to develop prototypes has been very valuable in this organization.

Their use of prototyping techniques also excludes the use of Excelerator. Excelerator's data dictionary is not compatible with the ADABAS data dictionary. There is now an interface available in the market, but this organization has not acquired it. Consequently, IS professionals do not want to do the same job twice; that is, entering data to both data dictionaries. Besides, since prototyping has proven to be very successful for users, Excelerator is only used for drawing data flow diagrams. ADABAS is used for data

specifications and it is combined with NATURAL for prototyping purposes.

Related to this hypothesis is the interesting issue of whether CASE tools were fully integrated in the IS development process. To this respect, answers were different. What is important here is to compare the introduction date to the integration date, if any. A summary of these dates follows.

Introduction date	Integration date
1986	Not yet
1988	1990-91
1987-88	1988
Mid-1987	End of 1987
March 1988	Not known
July 1987	July-August 1988
1989	Not known
1986	Never
1986	Not yet

In organization B, where the IS architecture is centralized, the perception of integration varied. The interviewees gave different dates for this event. For one interviewee integration took less than a year, for the other, it took from 2 to 3 years.

In organization C, where the CASE tool was introduced to the IS units in different dates, the perception of a date for integration also varied. For some that date is not

known, for another that event has not happened, and another believed that integration will never happen. When a date was given, the period from the introduction of the tool to the integration did not take more than a year.

In organization A, which has a decentralized architecture and where the tool was introduced in 1986, the integration to the IS development process has not been reached. That is, almost 6 years have passed without having the tool integrated in the development process.

Other Findings

1. CASE tool learning-curve-length.

Interesting answers were obtained not only when asking for a date for integration of the tool, but also when asking the length of the learning curve to use the tool.

The results show that training for using the tool is given mainly to systems analysts. It is evident from the results that different tools with different levels of complexity have different learning-curve lengths. That is not surprising. What is worth noticing is that the learning curve of the same tool is perceived as taking different lengths according to who answered this question even in the same organization. For example, one interviewee did not know how much time it takes to use the tool, another interviewee thought that it takes two years, while other respondents assigned a period ranging from 1 to 3 months to

the learning-curve-length.

The average years of experience, in these cases, was very similar. The average years of experience in structured methodologies was not that similar. But, in each case, more than 50% of the systems analysts have been trained in the use of the tool.

The possible explanation for these variations in the learning curve is that the meaning of mastering the tool is different for everybody. For some, the learning curve is the time to learn the mechanics of the tool. For others, it includes the time to learn the tool mechanics, and the time to use the tool during a systems project. That is, to use the tool to build a real system. Other factors can play an important role in the perception of the learning-curve-length. Systems complexity, the extent to which the tool is used, and of course, experience in structured methodologies can affect the pace of the developers learning process. Besides, no indication was found of how tool expertise was shared among developers.

2. Absence of standards.

This is an old and very familiar story in IS development. According to the survey results, there are no standards for types of application. Only one of the respondents said that the organization uses standards such as time and cost per type of project. What is worth mentioning is that not all IS professionals in this

organization, which has a centralized IS department, use standards. That is, there is not a standard uniformly followed across the organization.

3. CASE tools vs. CASE integration.

As presented earlier, one of the organizations surveyed has used more than one CASE tool in some of its IS units. Studying the relationship between CASE introduction and CASE integration, it becomes clear that something went wrong.

Only 33% of the interviewees thought integration had taken place. Only one IS professional of the remaining 67% thought that the primary emphasis on CASE usage had been to improve analysis quality. This comment could explain why integration has been difficult to achieve; many could think CASE is only a graphics or documentation tool. Of course, the same factors affecting the learning curve could be applied here to explain why integration is not being achieved. Also, IS management may not have enforced the importance of this kind of tool as a change agent in the systems development process.

4. Systems costs estimation.

Expert judgement is widely used in two of the surveyed organizations as a means to estimate the cost of a new system. IS professional's experience plays a very important role in costs estimation.

In the other organization, the CASE tool provides a kind of methodology to estimate the costs. Technical

aspects such as number of procedures in the system, and non-technical aspects such as working days and holidays are considered. After that, expert judgement is used to adjust the costs.

5. Success systems.

When IS professionals were asked to define when a system is considered to be a success, expressions such as "when the users like it" were obtained.

Only one of the interviewees emphasized the role of users when formally defining a successful system. The procedure followed in his IS unit is to obtain a written approval from the users stating that they are satisfied with the system. Only after that is the system released. In the event that users are not satisfied with the resulting system, the developers should continue working on the system until users approve it.

CHAPTER VI

CONCLUSION

Case study results show examples of organizations dealing with this technology and achieving different levels of benefits. In addition, the results reinforce the idea that technology by itself does not produce miracles in solving old problems in the IS field. The literature review presented in Chapter II and especially in Chapter III was done with the purpose of comparing CASE productivity issues from the survey. Unfortunately, some comparisons are impossible because the organizations under study did not have productivity measures in place. Thus, it is impossible to decide, for example, if one productivity measure is better than another.

According to the study, even when an organization acquires CASE for the right reasons, some aspects of its implementation are left out of the scenario. Standards, development and implementation of a program to record and evaluate the effectiveness and efficiency of the IS activities, are only two examples of what is ignored or underestimated. As the literature says and the survey confirms, the absence of historical data does not allow

comparison if IS activities are carried out better with this technology.

We have seen how, in at least two of the three organizations studied, the acquisition of CASE was a strategic decision. We have also found out that effective steps, such as training and pilot projects, were taken to ensure success when introducing CASE. What is not seen is the same commitment for evaluating the effect of this technology.

Organizations should assign resources to evaluate the effects of introducing CASE technology and discover what IS areas are not being improved and why. For example, some tools are based on a specific methodology. Without enough experience in that methodology CASE is not going to make a systems analyst better in his/her job. In this case, training in the methodology is more important than training in tool mechanics. More specifically, organizations should determine what area of IS development needs or lacks the appropriate emphasis before incorporating a CASE tool into the process. In other words, the tool by itself is not what needs to be emphasized. Rather, more significant problems should be addressed.

During the survey a feeling of "improving analysis quality" was perceived as the main benefit from CASE usage. But, again, there is no evidence -statistics or records- to support it. Feelings are not measurable, they are opinions

based on particular experiences. Solid evidence is what is needed.

One of the organizations surveyed acquired a CASE tool as the result of a suggestion from consultants developing a specific system. The consultants had experience using this tool, and they suggested it could facilitate the analysis of the system being developed. This organization had enough resources to acquire the tool. But the tool was not used after that development. One may wonder to what extent it is good for an organization to have resources to buy tools under these kind of circumstances. An organization with a tight budget cannot afford this luxury. Usually convincing reasons and probes are required to justify such an expense. This forces IS professionals to think about the benefits expected, and maybe they could conclude that it is not the right time for the tool, or it is not the right tool for the job. But, some thinking is required before buying.

A conclusion becomes evident from what has been mentioned. Something is being done to improve the developers work. However, IS management has failed to establish and enforce procedures to evaluate how IS activities are being conducted. In particular, IS management has failed to evaluate how this specific technology has affected the development of systems. But maybe this failure is related to what top management perceives as IS activities. As one of the interviewees

said: "They want quick, cheap and good systems. " Despite this situation, it is worth investing in CASE as a means to facilitate and standardize the way IS are developed, especially, as far as systems quality is concerned.

APPENDIX

COMPUTER AIDED SOFTWARE ENGINEERING (CASE) TOOLS AND PRODUCTIVITY

QUESTIONNAIRE

Introduction.

This questionnaire is part of a survey, the results of which will be used for the development of a information systems master thesis. The subject of the masters thesis is to study the extent, if any, of improved productivity in the information systems development through the use of CASE tools.

SPECIFIC RESPONSES WILL NOT BE ATTRIBUTED TO ANY ORGANIZATION IN PARTICULAR

I. ORGANIZATION DATA.

1. Name: _____

2. Location: _____

3. Number of employees at this location: _____ Number of employees in the organization: _____

4. Hardware Infrastructure.

	Model	Quantity
Mainframes	_____	____
	_____	____
	_____	____
	_____	____
Minicomputers	_____	____
	_____	____
	_____	____
	_____	____
Microcomputers	_____	____
	_____	____
	_____	____
	_____	____

5. Software Infrastructure.

Operating Systems	_____
	_____
	_____
	_____
Data Base Management Systems	_____
	_____
	_____
	_____
Programming Languages	_____
	_____
	_____
	_____
	_____

Spreadsheets _____

Other:
Please do not include
CASE tools here. _____

CASE Tools
Check all that apply:

	Product-Company Name	Introduction Date
[ ]	AnaTool - Advanced Logical	_____
[ ]	Analyst/Designer Toolkit-Yourdon, Inc.	_____
[ ]	APS-Sage Software, Inc.	_____
[ ]	AutoCode - Integrated Systems, Inc.	_____
[ ]	CASE Designer, CASE Dictionary - Oracle	_____
[ ]	Chen Toolkit - Chen & Associates, Inc.	_____
[ ]	COBOL/2 Workbench - Micro Focus, Inc.	_____
[ ]	DesignAid - RTrace	_____
[ ]	Excelerator - Index Technology Corp.	_____
[ ]	Foundation - Andersen Consulting Corp.	_____
[ ]	IDMS/Architect - Cullinet Software, Inc.	_____
[ ]	Information Engineering Facility - Texas Instruments	_____
[ ]	Information Engineering Workbench - KnowledgeWare	_____
[ ]	Life Cycle Productivity System - American Management Systems, Inc.	_____
[ ]	Path Vu or Retrofit - Peat Marwick Advanced Technology	_____
[ ]	Project Workbench - Applied Business Technology	_____
[ ]	SuperCASE - Advanced Technology International	_____
[ ]	Teamwork - Cadre Technologies	_____

[ ] Other

_____ _____
_____ _____
_____ _____
_____ _____
_____ _____

II. INFORMATION SYSTEMS ACTIVITIES.

1. How would you classify the information systems architecture in your organization?

   [ ] Centralized
   [ ] Decentralized
   [ ] Client/server
   [ ] Distributed processing

   Comments: _____
   _____
   _____
   _____

2. Information Systems Personnel.

		Analysts	Programmers
a.	Number	_____	_____
b.	Average years of experience	_____	_____
c.	Years of experience in structured methodologies (DeMarco, Yourdon, etc.)	_____	_____
d.	Number trained in CASE tools currently in use	_____	_____
e.	In your opinion, how long has the learning curve been in the use of CASE Tools (Years, Months, Weeks)	_____	_____

3. Information Systems Development.

   a. Is there an in-house developed methodology for building systems?

      [ ] Yes [ ] No

   Name of the methodology used: _____
   _____

   b. Check the methodologies used in building systems:

      [ ] De Marco structured analysis

      [ ] Gane/Sarson structured analysis

      [ ] Yourdon structured design

[ ] Jackson structured design

[ ] Ward/Mellor (real-time systems)

[ ] Martin information engineering

[ ] Object-oriented

[ ] Other. Please specify _____

_____

c. Applications developed:

Types:

[ ] Basic (Payroll, Inventory, Accounting, etc.)
[ ] Technical/Scientific
[ ] Other

Processing:

[ ] Batch [ ] Interactive/Real Time

Comments: _____

d. Does the IS Department have standards for each type of application?

[ ] Yes [ ] No
[ ] Cost
[ ] Time
[ ] Number of people assigned to the project

Comments: _____

e. Are there statistics on applications development costs/time?

[ ] Yes [ ] No (If No, go to f.)

If Yes,
Is there a difference between applications developed using CASE and those developed without it?

[ ] Yes [ ] No
[ ] Increasing costs/time?
[ ] Decreasing costs/time?

Comments: _____

f. Are there statistics on maintenance of existing information systems?

[] Yes [] No (If no, go to g.)

If Yes,
Is there a difference in the number of requests for maintenance between those systems developed with CASE and those developed without it?

[] Yes [] No
[] Larger number of requests for systems using CASE
[] Larger number of request for systems without CASE

Comments:_____
_____
_____
_____

g. Has a change occurred in the information systems backlog after the introduction of CASE tools?

[] Yes [] No
[] Increase
[] Decrease

Comments: _____
_____
_____
_____

h. Does your organization use consulting companies for systems development?

[] Yes, for systems developed WITHOUT CASE tools
[] Yes, for systems developed WITH CASE tools
[] No

Comments: _____
_____
_____
_____

4. Productivity Issues.

a. Does the IS Dept. have a definition for productivity?

[] Yes [] No (If No, go to d.)

If Yes, please specify:
_____
_____
_____
_____

b. When was this definition established?

_____ [] Not known

c. Did the IS Dept. use another definition prior to the specified above?

[] Yes [] No

If Yes,
When was that definition established?
_____ [] Not known

d. Are there any measurements for productivity?

[] Yes [] No (If No, go to 5)

e. Types of productivity measurements used during:

Analysis Phase _____
_____
_____
_____

Design Phase _____
_____
_____
_____

Programming Phase _____
_____
_____
_____

Comments: _____
_____
_____
_____

5. Introduction of CASE tools.
In the organization.

a. Approximate date when the first CASE tool was introduced:

_____ [] Not known

b. Name of the first CASE tool introduced: _____
_____

c. Was the introduction of CASE tools part of the information systems strategic plan?

[] Yes [] No [] Not known

d. Did your organization use a consulting company for assistance in implementing CASE tools?

[] Yes [] No [] Not Known

Comments: _____
_____
_____
_____

In the development of information systems.

e. Was a plan developed and followed to introduce the use of CASE tools?

[] Yes (If Yes, go to f.) [] No

If No,
Please describe how these tools were integrated to the information systems development process.

_____
_____
_____
_____
_____
_____

f. Approximate date when CASE tools were fully integrated to the information systems development process.

_____ [] Not known

g. Please indicate the name, if any, of other departments, besides the IS Dept., using CASE tools.

_____
_____

Comments: _____
_____
_____
_____

h. Please indicate the name of analysts or programmers that have worked or are currently working with CASE tools in your organization who might be available for future interviews.

NAME	DEPARTMENT
_____	_____
_____	_____
_____	_____
_____	_____

Would you be available for a follow-up interview? If Yes,

Name: _____

Job Title: _____

Department: _____

Telephone: _____

Thanks for taking the time to fill out this questionnaire. Data provided here is crucial for the completion of the study on CASE tools and productivity.

Date: _____

BIBLIOGRAPHY.

Addelston, Jonathan D. "Software Engineering Perspectives," Technology Transfer. April 1990. pp. 12-15.

Anthes, Gary H. "User Role Gains CIO Backing," Computerworld. February 24, 1992. p. 63

Boone, Greg and Vaughan, Merlyn. "Getting the Process Right," Computerworld. June 10, 1991. pp.75-77.

Bouldin, Barbara M. "What Are You Measuring? Why Are You Measuring It?," Software Magazine. August 1989. pp. 30-39.

Braithwaite, Kenmore. "Choosing CASE Tools," DBMS. January 1991. pp. 44-51.

Brathwaithe, Kenmore S. Applications Development Using CASE Tools. San Diego, California: Academic Press, Inc., 1990.

Bryant, John. "The problem with CASE," Systems International. April 1990. pp. 75-76.

Burke, John P. "Though CASE," HP Professional. July 1991. pp. 30-39.

Bush, Eric. "CASE for Existing Systems," InfoStrategy: The Executive's Journal. Spring 1991, pp. 31-39.

Case, Albert F. "Information Engineering and CASE Environments - Part One," The CASE Report. November 1987. pp. 1-3.

"CASE Tool Roundup", DBMS. July 1991, pp. 62-69.

"CASE Use Is Growing Up, but in Surprising Ways," Datamation. May 1, 1992. pp. 108-109.

Chikofsky, E.J. Software Development Computer Aided Software Engineering (CASE). Washington, DC: IEEE Computer Society Press, 1989.

_____"Making CASE Pay Off," CASE Directions. Vol. 1, No. 1. pp. 14-16.

Costello, Jim. "Radical Ways to Help Projects; How Does a Software Developer Avoid Boredom and Speed up Project Times?", Computer Weekly, June 20, 1991, p. 30.

El Louadi, Mohammed, Pollalis, Yannis A. and James T.C. Teng. "Selecting a Systems Development Methodology", Information Resources Management Journal. Winter 1991, pp. 11-19.

Evans, Michael W. The Software Factory: A Fourth Generation Software Engineering Environment. New York: John Wiley & Sons, Inc. 1989.

Fersko-Weiss, Henry. "CASE Tools for Designing your Applications", PC Magazine. January 30, 1990. pp. 213-251.

Forte, Gene and Norman, Ronald J. "A Self-Assessment by the Software Engineering Comunity," Communications of the ACM. May 1992. pp. 28-32.

Fisher, Alan S. CASE: Using Software Development Tools. New York: John Wiley & Sons, Inc., 1991.

Francis, Ted. "CASE Speeds Systems Development, Increases Programmer Productivity", Bank Administration. October 1988. pp. 82, 84.

Freedman, David H. "In Search of Productivity", Infosystems. November 1986. pp. 12,14.

Garvey, Martin. "Racing for CASE Standards; Lacking a Formal Standard, Users Seek to Integrate CASE Tools -and Vendors Oblige", Information Week. June 17, 1991. p. 54.

Geller, Rob. "Structured for Success," InformationWeek. April 6, 1992. p. 70.

Gibson, Michael Lucas. "The CASE Philosophy", Byte. April 1989. pp. 209-218.

Gibson, Michael L.; Snyder Charles A., and Houston H. Carr. "Why CASE Belongs to Strategic Business Management", InfoStrategy: The Executive's Journal. Winter 1991. pp. 17-23.

Gibson, Michael L.; Snyder, Charles A., and R. Kelly Jr. "CASE: Clarifying Common Misconceptions", Journal of Systems Management. May 1989. pp. 12-19.

Graham, Carol. "CASE Cracks Applications Backlog", Datamation. March 15, 1991. pp. 97-99.

Hayley, Katheryn J. and Lyman, H. Thaine. "The Realities of CASE", Journal of Information Systems Management. Summer 1990. pp. 18-23.

Hubbard, Craig. "Increased Productivity Isn't Always Number One", Computing Canada. May 24, 1990. pp. 29,32.

Huff, Clifford C. "Elements of a Realistic CASE Tool Adoption Budget," Communications of the ACM. April 1992. pp. 45-54.

Humphrey, Watts S. "Characterizing the Software Process: A Maturity Framework", IEEE Software. March 1988. pp. 73-79.

Jones, Capers. "How not to Measure Programming Productivity", Computerworld. January 13, 1986. pp. 65-76.

Keyes, Jessica. "Gather a Baseline to Assess CASE Impact", Software Magazine. August 1990. pp. 30-43.

Knight, Robert. "Experts Stress 'Process'; Users Find that CASE Goes Far Beyond Tools", Software Magazine. Jun 1991. p. 34.

Kolman, Joe. "Information Management: Getting Down to CASEs", Institutional Investor. Aug. 1990. pp. 119-122.

Loh, Marcus and Nelson, R. Ryan. "Reaping CASE Harvests", Datamation. July 1, 1989. pp. 31-34.

"Making Case for CASE", Byte. December 1989. pp. 155-171.

Maria, Arturo. "CASE Technology: Today's Reality", Journal of Systems Management. February 1991. pp. 18, 26.

McClure, Carma. CASE is Software Automation. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

McClure, Carma. "The CASE Experience", Byte. April 1989. pp. 235-244.

Mosley, Daniel J. "Getting CASE Straight: Can You Use It or Not?", Chief Information Officer Journal. Fall 1990. pp. 55-59.

Necco, Charles R.; Tsai, Nancy W. and Kreg W. Holgeson. "Current Usage of CASE Software", Journal of Systems Management. May 1989. pp. 6-11.

Ng, Peter A. and Yeh, Raymond T. Modern Software Engineering: Foundations and Current Perspectives. New York: Van Nostrand Reinhold, 1990.

Pastore, Richard. "Proving the Case", Chief Information Officer Journal. October 1, 1991. pp. 28-38.

Perry, William E. "Make an Investment in Your ADP Workers," Government Computer News. October 14, 1991. p. 86.

QED Information Sciences. CASE- The Potential and the Pitfalls. Wellesly, Massachusetts, 1989.

Radding, Alan. "Programmer Productivity: No Giant Leaps", Computerworld. August 5, 1991. pp. 59-60.

Ray, Gary. "Better CASE Translation," Computerworld. August 10, 1992. p. 61.

Ridgway, Helen. "CASE: The Wonder Cure", IBM System User. May 1991. pp. 17-18.

Rochester, Jack B. "Improving Application Development Productivity", I/S Analyzer. March 9, 1990. pp. 1-12.

Santosus, David. "The Fine Art of Figuring CASE Payback", Computerworld. Mar 27, 1989. pp. 74-75.

Schindler, Max. Computer-Aided Software Design: Build Quality Software With CASE. New York: John Wiley & Sons, 1990.

Scudder, Richard A. and Kucic, A. Ronald. "Productivity Measures for Information Systems", Information & Management. May 1991. pp. 343-354.

Slater, Derek. "PacBase, IEF Lead Rising CASE Satisfaction," Computerworld. July 20, 1992. pp. 81-82.

Smith, Al. "No Measure, No Change", Computerworld. October 8, 1990. p. 70.

Souza, Eileen. "The Impact of CASE on Software Development", Journal of Information Systems Management. Winter 1991. pp. 17-24.

Stamps, David. "CASE: Cranking Out Productivity", Datamation. July 1, 1987. pp. 55-58.

Statland, Norman. "Payoffs Down the Pike: A CASE Study", Datamation. April 1, 1989. pp. 32-33, 52.

Vessey, Iris; Jarvenpaa, Sirkka L., and Noam Tractinsky. "Evaluation of Vendor Products: CASE Tools as Methodology Companions," Communications of the ACM. April 1992. pp. 91-107.

Weiss, Ray. "Practical CASE Makes Strides", Electronic Engineering Times. June 3, 1991. pp. 41,42.

White, Leonard R. "Function Point Analysis: The Manager's Tool", Chief Information Officer Journal. Fall 1990. pp. 60-64.

Wilder, Clinton. "Systems Development Stalled; Survey Respondents Find Frustration with Old Code, Obsolete Skills", Computerworld. June 17, 1991. p. 4.

Yellen, Richard E. "What Do Users Really Think about CASE," Journal of Systems Management. February 1992. pp. 16-17.