

**A Computer-Aided Software Engineering (CASE)
Approach
to
Business Process Reengineering (BPR)**

by

Faiza I. Khan

A Thesis Submitted to the
Graduate Studies Office
in Partial Fulfillment of the Requirements for the Degree of
Master of Business Administration in Aviation

Embry-Riddle Aeronautical University
Daytona Beach, Florida

Fall 1995

UMI Number: EP31945

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform EP31945
Copyright 2011 by ProQuest LLC
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

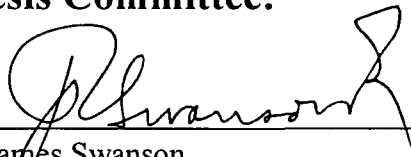
ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

**A Computer-Aided Software Engineering (CASE)
Approach
to
Business Process Reengineering (BPR)**

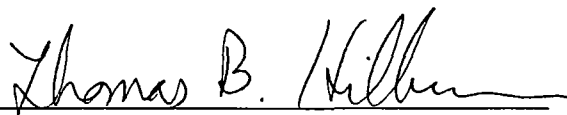
by
Faiza I. Khan

This thesis was prepared under the direction of the candidate's thesis committee chairman, Dr. James Swanson, Aviation Business Administration Department, and has been approved by the members of his thesis committee. It was submitted to the Aviation Business Administration Department and was accepted in partial fulfillment of the requirements for the degree of Master of Business Administration in Aviation.

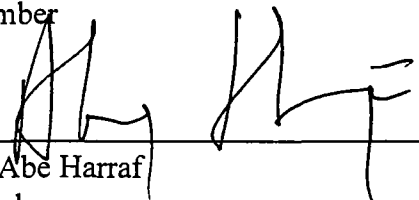
Thesis Committee:



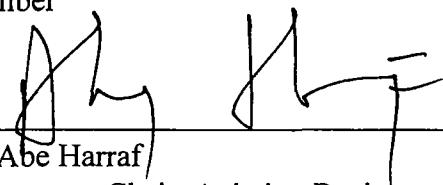
Dr. James Swanson
Chairman



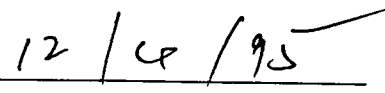
Dr. Thomas Hilburn
Member



Dr. Abe Harraf
Member



Dr. Abe Harraf
Department Chair, Aviation Business Administration



Date

ACKNOWLEDGMENTS

The author wishes to express special thanks to the Thesis Chairman, Dr. Swanson, whose constant encouragement, helpful counsel and practical suggestions were crucial to the successful outcome of this thesis. Appreciation is also due to Dr. Harraf and Dr. Hilburn, Thesis Committee Members, for their assistance in preparing this manuscript.

ABSTRACT

Author: Faiza I. Khan
Title: A Computer-Aided Software Engineering (CASE) Approach to Business Process Reengineering (BPR)
Institution: Embry-Riddle Aeronautical University
Degree: Master of Business Administration in Aviation
Year: 1995

This thesis addresses the problem of lack of procedures by which managers can identify business processes that would benefit from reengineering and which could be used to guide the BPR activity to its successful conclusion. The purpose of this study is to analyze various Software Engineering (SE) principles to determine whether they could be used to support Business Process Reengineering (BPR) activities either directly or with some modification, and if so, whether available Computer-Aided Software Engineering (CASE) tools might be applied to BPR in support of managers who wish to implement it. The principles of SE and BPR are discussed and the extent to which CASE tools can comply with BPR's effort of revising and eliminating irrelevant standards and processes is analyzed. The research showed that SE and BPR have several, but not all, principles in common and thus, certain CASE tools can be utilized in BPR process. The study suggested the next step in BPR research be toward coming up with procedures to help implement BPR in an organization.

TABLE OF CONTENTS

	page no.
ACKNOWLEDGMENTS	iii
ABSTRACT	iv
LIST OF TABLES	vii
<i>chapter</i>	
1. INTRODUCTION	1
<i>Problem Statement</i>	2
<i>Previous Relevant Research</i>	2
<i>Research on Software Engineering</i>	4
<i>Research on CASE Tools</i>	6
<i>What is CASE?</i>	6
<i>CASE Classification</i>	9
<i>Key Components of CASE Products</i>	13
<i>Research on BPR</i>	19
<i>Research on Application of Software Tools to BPR</i>	20
2. METHODS DESCRIPTION	23
3. ANALYSIS: CASE TOOLS & BPR	26
<i>Business Process Reengineering</i>	26
Definition	26
Features of Reengineered Business Processes	27

<i>Software Engineering</i>	32
Definition	32
Features of Software Engineering Processes	32
<i>Applying CASE Tools & SE to BPR</i>	34
4. CONCLUSION AND RECOMMENDATIONS	40
REFERENCES	43

LIST OF TABLES

		page no.
Table 1	A List of Selected SE Principles	23
Table 2	A List of Selected BPR Principles	24

CHAPTER 1

INTRODUCTION

Competition, customer demand, federal legislations, globalization, and the threat of takeovers has made organizations all over the world think about trying to survive through rapidly evolving technologies, new advances in databases, networking and telecommunications, and modern management tools. One of these new management tools is called Business Process Reengineering (BPR). BPR has emerged as a major practice in improving the productivity and efficiency of today's business enterprises. Unfortunately, however, BPR has become something of a blurred concept in the U. S., and it has attained "buzzword" status. Managers who have not "lived through" the process have little guidance as to how to implement BPR effectively.

Software Engineering (SE) has developed as a support mechanism for people engaged in the relatively technical task of analyzing business systems and producing computer software to carry out complex tasks involved in those systems. A number of principles of SE have been established, which, when applied to the various tasks involved, assist in their accurate and timely completion.

The purpose of the research described in this thesis is to examine appropriate SE principles to determine whether they could be used to support BPR activities either directly

or with some modification, and if so, whether available CASE tools might be applied to BPR in support of managers who wish to implement it.

The subjects of analysis of the thesis are SE principles, as embodied in CASE tools, and their potential applications to BPR. SE is an approach to engineering software systems whereas BPR is an approach to engineering business processes. This study will show that even though CASE is a set of tools for SE, it can serve as a tool for BPR as well. The features of BPR and SE, in general, will be discussed to explore the extent to which CASE tools can help the user-designers in BPR develop and design (or redesign) their individual processes while still complying with organizational standards. During this analysis, CASE tools will also be analyzed to find out the extent to which they comply with BPR's purpose of revising and eliminating irrelevant standards and processes.

Problem Statement:

The problem addressed by the research reported in this thesis is the lack of procedures by which managers can identify business processes that would benefit from reengineering, and which can be used to guide the BPR activity to its conclusion.

Previous Relevant Research:

Even though designers of information systems have historically been the last to apply computer-based tools to improve the quality, reliability, and productivity of their own work, ... this tradition is changing as organizations

involved in both commercial data processing and embedded real-time applications review their heavy investment in software and information systems (Chikofsky and Rubenstein, 1988, p.11).

Now, these information system designers are gradually realizing that they can achieve greater system development productivity while providing a new and reliable approach to reengineering information systems through CASE and other software tools. Fuggetta (1993, p.25) observed that "such software and CASE tools can help improve complex and expensive activities like the design, implementation, delivery, and maintenance of software in today's software development environment".

While much research has been conducted on tools for software engineering, many claims are being made for and about BPR (Parker, 1993). Cavanaugh (1994, p.7) quoted Michael Green to the effect that BPR is about "rethinking and redefining followed by a restructuring or reorganizing of all the work that [we are] trying to do". For the past few years, many institutions have been forced into reactive patterns of dealing with change due to various dynamic influences. The impact of change seems dramatic with different degrees of success. Hammer and Champy (1993) said that BPR can help existing organizations reinvent themselves while changing the whole organizational structure. According to them, "despite the prominent role played by information technology in business reengineering, it should now be clear that reengineering is not the same as automation. Automating existing processes with information technology is analogous to paving cow paths" (p.48). They further stated (p.101) that IS can play the role of an enabler but not of a driver. "As an essential enabler in reengineering, modern information technology has an importance to the

reengineering process that is difficult to overstate. But companies need to beware of thinking that technology is the only essential element in reengineering". Hammer used the term "enabler" because "it permits companies to reengineer business processes," and "driver" because "it does not lead the change" (Maglitta, 1994, p.83). Hammer also said that "many IS people still too quickly confuse business reengineering with software engineering. Too many people still think the answer is CASE tools. We need to totally rethink how systems are developed, starting from scratch" (p.84).

Research On Software Engineering:

Businesses around the world depend more and more on software in the very basics of their operations. Quality of software design and quality of business service are very much linked. Customers' needs are served better if the quality of software being used is better. Quality of software depends heavily on the software engineering of the process. "Software Engineering is the systematic development, operation, maintenance, and retirement of software" (Conger, 1994, p.1).

Software is the sequence of instructions in one or more programming languages that comprise a computer application to automate some business function. Engineering is the use of tools and techniques in problem solving. Putting the two words together, software engineering is the systematic application of tools and techniques in the development of computer-based applications (p.2).

In order to achieve a quality product, quality of process is desirable. "The software engineering process describes the steps it takes to develop the system" (p.3). A SE process is more effective and efficient if certain tools are applied during the process. One of these tools is CASE tools. However, many companies fail at applying CASE tools and thus, achieving efficiency in their processes mostly due to neglecting basic SE principles even though they select the best tools, techniques, and methodologies.

These basic principles are: effectiveness, quality, rigor, process improvement, and automation.

Effectiveness is building the right system, the right way, right from the start.

Quality should first be defined by the users, the maintainers, and management.

It might mean error-free code, meeting expectations, or meeting specifications depending on who is defining it.

Rigor has been defined by Vaughn Merlyn, as correctness, consistency,

coherency, and completeness of a process and the resultant deliverables.

Process improvement implies continuously improving one's processes to improve quality and effectiveness. The focus is on process and not on product.

Automation includes the mechanization of the existing task (Flecher and Hunt, 1993, p.37).

Success in any field is marked by a measurable end point. Unfortunately, software developers lagged far behind other professionals in establishing standard metrics. However, a lot is being done recently in this aspect of applying software metrics to measure success in this field.

Research on CASE Tools:

Software engineering is an emerging discipline with a goal of producing reliable software products cost-effectively. It does this with a number of technologies, including structured techniques and automated development systems. Its various elements have been developed independently for the past 25 years, and are now being brought together as a cohesive approach in the last few years in the form of CASE. "CASE makes the transition from the world of individual, unique approaches in software development to a disciplined, structured, standardized world of software engineering" (QED Information Sciences, Inc., 1989, p.34).

What is CASE?

CASE is one of the technologies to design, implement, deliver, and maintain those software activities which are not only difficult to control but also complex and expensive. The need for different approaches to satisfy the needs of different industrial customers has resulted in evolution of several CASE tools. This availability of a large number of such CASE tools, on one hand, has contributed to the improvement of SE practice, but on the other, it has created critical problems regarding its definition and classification. Sodhi said that, "Computer-Aided Software Engineering (CASE) encompasses a collection of automated tools and methods that assist software engineering in the phases of the software development life cycle" (1991, p.7). Sommerville defined CASE as "the term for software tool support for the software engineering process" (1992, p.167). Pressman extended the scope of CASE when he said that, "The workshop for

software engineering is called an integrated project support environment, and the toolset that fills the workshop is CASE" (1992, p.131). Fuggetta (1993) first classified CASE into tools, workbenches, and environments and then proposed different definitions for each classification. He defined CASE tool as "a software component supporting a specific task in the software production process" (p.35). Workbenches, according to him, are "integration software-process activities in a single application" (p.36). Environments, in turn, are "a collection of tools and workbenches that support the software process" (p.37). Each of these classifications have several subparts.

CASE "emphasizes structured methods, with defined and standardized procedures" (QED Information Sciences, Inc., 1989, p.76). Chikofsky and Rubenstein said that "A CASE environment provides the analyst or systems developer with facilities for drawing a system's architectural diagrams, describing and defining functional and data objects, identifying relationships between system components, and providing annotations to aid project management" (1988, p.3).

CASE technologies are engineering technologies. Engineering is defined by Webster's as 'the application of science and mathematics by which the properties of matter and the sources of energy in nature are made useful to man in structures, machines, products, systems, and processes.' This is exactly analogous to CASE in information systems (QED Information Sciences, Inc., 1989, p.53).

CASE was created because of the increased pressure on applications systems developers due to the lack of disciplined, standardized methods for managing the

applications systems development process. The essence of all CASE technologies is that they have a structured approach. They provide a way of attacking systems development with well-defined methods and standardized procedures. The key gain in most organizations has been the quality of the systems being developed because software developers can detect errors and inconsistencies and refine specifications easily to reflect their own customers' needs using different CASE tools. "The purpose of the CASE is not simply to increase programmer productivity with a number of tools. The purpose of CASE is to achieve an overall increase in productivity among the programmers, analysts, users, and managers together" (QED Information Sciences, Inc., 1989, p.71).

The user's various work products which are stored in an integrated non-redundant form in a central repository or dictionary on the workstations or on a central server or host system can be used to analyze the collected information for the detection of inconsistencies and errors early in the software life cycle.

A software system can be defined as an organized collection of data, machines, procedures, documents, and other entities that interact with each other and their environments to attain a desired goal. That is why software engineering is only part of the whole picture. Software engineering obviously handles the automated procedures, but it must also consider the manual procedures, the data, the human interface, and the organization of the network of machines to support the whole. The overall thrust of software engineering includes all aspects of the system that can be run centered on the computers. Technologies such as CASE are of great use when they are

integrated into a whole system rather than standing alone (QED Information Sciences, Inc., 1989, p.35).

Since customers have become interested only in systems that perform the procedures they need, information systems professionals have been working out how to apply technology to bring organized automation to the current aspects of the systems desired by the users. Users have been asked to "fit themselves in" to the realistic bounds of the available technology, and often to change their requirements to adapt to the operation that can be made available. CASE technologies are allowing us to systematically rethink the "system" from one with specific program limits to one that approaches users' desires.

The CASE tools provide for the use of data flow diagrams, structure charts, entity relationship diagrams, logical data models, and presentation graphics. The most important feature of a well developed CASE environment is adaptability. The environment must be able to operate on many hardware configurations in a consistent and user-friendly manner. Organizations using CASE with adaptability have experienced various degrees of improvement in productivity (Chikofsky and Rubenstein, 1988, p.13).

CASE Classification

Since many packages of CASE tools have been introduced for different levels of software, software engineers and researchers have classified them by function or by lifecycle. However, such classifications do not cover all available CASE tool offerings.

Classification by Function:

One way of classifying CASE tools is according to the functions they perform in developing a software (QED Information Sciences, Inc., 1989). Three functions to which CASE tools can be applied are:

- (1) In analyzing the areas that are difficult to handle manually,
- (2) In assisting the management and accomplishment of the CASE process, and
- (3) In the automation of previously manual tasks.

These three functions might or might not be mutually exclusive. The first function deals with the analysis phase of the software development lifecycle process. Analysis in areas that are difficult to handle manually can be made easier with the use of certain CASE tools. The assistance capabilities of such tools depend heavily upon the capabilities of the CASE tool developer. The second function deals with the areas where assistance in managing and accomplishing the CASE process is needed. It includes tools which can accumulate and track information or make information available to other CASE tools and systems in an organized way. These tools can prove to be of extreme value to the software engineers since they increase efficiency and decrease the staff efforts. They help in controlling the systems development process or aid in the handling of data that is used in doing so. The third function is the automation of certain procedures which eliminates some manual activities and makes the rest more accurate. In addition, such automation CASE tools provide matrices of information that have been prepared in one phase to be used in another.

Classification by Life Cycle Phase:

Through this classification, CASE products are classified according to the different phases of the SDLC in which they can be useful (QED Information Sciences, Inc., 1989). A CASE tool may be of use in one or more of the phases of the life cycle. This type of classification bears the advantage of being easily justified to management since it is related to particular phases of the SDLC. Under this classification, CASE tools are subdivided into Upper CASE or Lower CASE. Upper CASE tools, which are the most common in the market today, are used in the initial activities of software development. They play a critical role in getting the process started since they deal with the requirements phase involving communication between the analyst and the user. They are capable of transporting or transforming data between the various tools, are generic, and can be used with a variety of other tools. Lower CASE tools, on the other hand, help in design, coding, testing, implementing, and maintaining the system.

Classification by Production Process:

Fuggetta (1993, p.28) classified CASE products by dividing the whole software engineering process into three technologies: production process, meta process, and enabling technologies. The production process includes all activities, rules, methodologies, organizational structures, and tools used to conceive, design, develop, deliver, and maintain a software product. The purpose of the meta process is to acquire and exploit new products supporting software-production activities. It also helps in the improvement and innovation of the procedures, rules, and technologies used to deliver the organization's artifacts. These

two technologies are supported by an infrastructure which can be implemented by enabling technology (operating system services and data base management).

CASE products used in a production process can be classified according to the breadth of support they offer. Fuggetta (p.29) has classified them into three categories:

- (1) Tools supporting only specific tasks in the software process
- (2) Workbenching supporting only one or few of the activities
- (3) Environments supporting a large part of the software process if not the whole software process.

Workbenches and environments are generally built as collections of tools which in turn are usually stand-alone products. Tools can be further classified into: editing, programming, verification and validation, configuration-management, metrics and measurement, project-management, and spreadsheet tools. Workbenches integrate in a single application several tools supporting specific software-process activities. They achieve a homogeneous and consistent interface, easy invocation of tools, and access to common data; and they can be further classified into business planning and modeling, analysis and design, user-interface development, programming, verification and validation, maintenance and reverse-engineering, configuration-management, and project-management workbenches. The third subpart of production process CASE is an environment which is a collection of tools and workbenches to support the software process. It has been classified into toolkits, integrated, language-centered, and fourth generation environments.

Classification by Techniques:

CASE tools can be categorized into the following classes according to the purpose they serve:

1. *Information Engineering Techniques* are the products that provide a knowledge base for the creation and maintenance of enterprise models, data models, and process models. Such repositories are usually derived from the strategic plans of the enterprise and are used to ensure that all processes and data definitions are consistent. Several models are used to create and maintain a data processing system within a clearly-defined life-cycle process.

2. *Structured Diagramming Techniques* are the products that support the basic structured diagrams of data flow, control flow, and entity-relationship in SE. Structured Development Aid Techniques are the products that provide the analysts with the aides to structure a SDLC.

3. *Application Code Generator Techniques* are the products that generate code in a programming language e.g. COBOL.

Key Components of CASE Products

Sommerville (1992, p.56) stated that the phases of the software development life cycle are: feasibility study, requirements analysis, system design, programming, testing and maintenance. CASE products are basically designed for the earlier phases of the SDLC. Figure 1 describes the key components of the CASE products currently in the market (QED Information Sciences, Inc., 1989, p. 201):

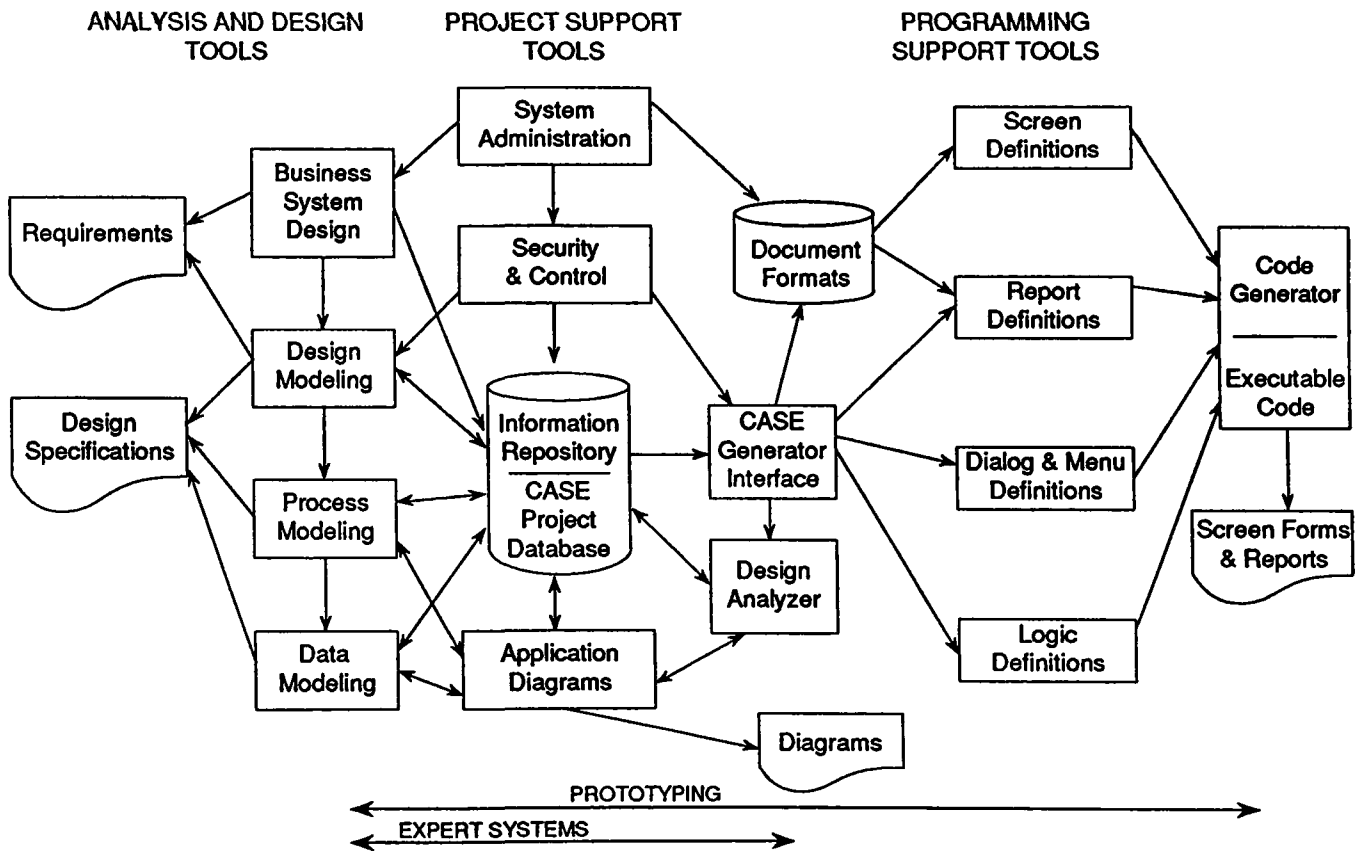


Figure 1 Key Components of CASE Tools

Systems Administration:

This component usually includes planning, budgeting, life cycle management, and monitoring, and consists of the project management functions of integrated CASE tools. It is fully capable of managing both the central information repository and the application diagrams and of handling such functions as version control, security control, and requirements alteration as well.

Business Systems Design:

This component is more of business modeling instead of system modeling and combines elements of planning, organizational structure and business analysis. It is difficult to structure this component completely because different business problems require different amounts of modeling and analysis.

Design Modeling:

This component moves from the business model to the system model and uses design automation techniques which are front-end, graphical design techniques. Such techniques are now becoming available in more and more products and specify system in graphical form analyzing the diagrams automatically for design errors and inconsistencies. Such automation tools incorporate all tools needed to go from the design specification to the code generator supplying information to the information repository en route.

Process Modeling:

Process modeling includes diagramming techniques such as data flow diagrams, structure charts, dialog flow, etc. which are at the heart of CASE methods. Such CASE products evaluate the diagrams and the text and can check for syntax, consistency and the

cohesive structure of the program. It creates the requirements definition, functional decomposition, program structure, and the program logic, and transmits them to the central information repository. Then, it transmits data flow diagrams and program structure charts to the application diagrams data base.

Such techniques let the software engineers and analysts draw diagrams of the system design and provides them with the flexibility to verify and revise such designs at their convenience. Computer-Aided Design (CAD) and other programming techniques allow the users to create and revise their drawings with a change in the requirements.

Data Modeling:

This component transmits information to the repository on the data entities, data relationships, and data elements. It transmits entity-relationship diagrams to the application diagrams database.

Security and Control:

Security provides necessary access and change control that might be needed and keeps account of accesses and changes. It is of critical importance but is usually given little attention. It includes:

Version Control,

Configuration Management,

Defect Tracking,

Requirements Tracing,

Dependency Management,

Code and Design Reuse, &

Access Control.

Central Information Repository:

The repository handles all information resources and the data dictionary. It includes functional models, data base description, program logic, screen and report definitions, program structures and change information. In addition, it includes all the administrative and security information.

Repositories can be called the "CASE tools' heart" since they are the knowledge base to store information about the whole organization. It usually holds complete information about an organization's structure, enterprise model, functions, procedures, data models, data entities, entity relationships, process models, etc. In addition to that, information about the procedure code of the CASE itself is also held there.

Application Diagrams:

This component handles all data flow diagrams, data structures diagrams, and program structure charts.

Document Formats:

This component contains complete information about all screen definitions and report definitions. It eliminates all further work on document definitions by maintaining documents standards and is a factor in the security of the program output.

CASE Generator Interface:

CASE generator interface is individual to any CASE tool developer and is a system element within any set of integrated CASE tools. It handles data from the CASE project

database and makes it available in the correct format for the code generator and the reporting programs.

Design Analyzers:

Sometimes system designs do not work due to internal consistencies and complications. Design Analyzers which are usually based on formal information models are used to detect internal inconsistencies in the design specifications.

Screen Definitions:

This component is responsible of receiving screen and report definitions from the central repository and transmits screen forms and reports to the output files.

Report Definitions:

This manages diagrams on the report layout that are used to create and format reports including the specifications of data items, page headings and footings, column headings, etc.

Dialog & Menu Definitions:

The storage and availability of programming for screen dialogs and menus is made possible by this component.

Logic Definitions:

Logic definitions store and make available the programming for logic definitions and logic handling in the reporting phase of the program.

Code Generators:

CASE tools can be used as a code generating tool if a tight coupling is present between the front end CASE tools and the back end code generator. It receives information on the functional model and the database description from the central repository. It receives

information from the application diagrams database and receives the sequence and layout of graphics and text from the document formats file and the definition files.

Expert Systems:

Expert System CASE tools use the knowledge stored in the repositories to make different inferences about an organization. Even though CASE tools with expert system capabilities are currently being used for very few purposes, there is a greater chance in the future of them being used at all phases of the SDLC.

Research on BPR:

A system is a set of interrelated components that must work together to achieve some common purpose. Four general key components of the organization that must work in concert for the whole organization to be effective are people, technology, tasks and organizational structure. BPR involves a thorough reassessment of these four components of an organization as a system -- people, procedures, organizations, and technologies (Martin et al., 1994, p.23).

Three forces, separately and in combination, are driving today's companies deeper and deeper into territory that most of the executives and managers find frighteningly unfamiliar. Hammer and Champy (1993, p.24) called these forces "the three Cs": Customers, Competition, and Change. According to them (p.51), reengineering involves the following principles for analyzing and reorganizing business as a system.

Organize business around outcomes, not tasks (i.e. one person should be able to perform all steps in a given process).

Assign those who use the output to perform the process (i.e. the person who is more interested in the result is accountable for the process).

Integrate information processing into the work that produces the information (i.e. information should be processed at its source).

Create a virtual enterprise by treating geographically distributed resources as though they were centralized (i.e. distinction between centralization and decentralization disappears).

Link parallel activities (i.e. parallel processes should be constantly coordinated rather than coordinating their results only).

Let the people who perform the work make all the decisions in order to build the control into the system from the process.

"The goal of BPR is to achieve an order of magnitude improvement, rather than incremental gains, from automating current processes" (Hammer and Champy, 1993, p.120).

Research on Application of Software Tools to BPR:

Business processes and software development processes are both information-intensive processes carried out by people with the support of computer applications, databases, and tools. The study of each is motivated by similar objectives, leading to a number of issues of mutual interest. Some of the shared

objectives are to facilitate communication (both within an organization or between an organization and its customers), to aid in training of staff, to make standardization possible (at the corporate, departmental, and/or project levels), and to establish a baseline for process improvement (Huff , 1993, p.99).

In addition to facilitating the communications within the organization, another objective for both BPR and SE is to automate the processes. They both are based on automating the process as a whole instead of automating individual steps therefore relying on process support software which can share some of the responsibilities in this process automation.

Two common properties of SE processes and BPR processes are that both of them are complex and involve parallel, interacting tasks. Process complexity is hidden from view when the process is described at a very high level of abstraction. However, this complexity can be handled by using multiple levels of abstraction and introducing different process phenomena at those levels.

CASE has basically been classified and recognized as a software engineering tool. However, it can be utilized as a tool for BPR. Hansen (1994) related BPR to software engineering by stating that business processes have many parameters to deal with and that this problem can be solved with much more ease if software tools are used for the purpose. Even though more expensive tools have been introduced in the past, cheaper and more advanced tools like Computer-Aided Process Re-engineering (CAPRE) are now on the market. These tools are being designed to handle both the functional and the behavioral view. Graphical interfaces and icons are used to represent actions, functions, and objects

(tangible and intangible things that flow through the process). According to Hansen , a minimum iconic block set should consist of operations, mathematical and logical operations, transactions, repositories, queues, decisions and events. In addition to an iconic block set, such tools should be able to provide its users with:

Process flow -- to show the flow of objects through a business process,

Discrete-event modeling -- to help vary the timings of the processes since they all usually don't happen at the same time,

Scenario analysis -- to build a different scenario, enter inputs in them and capture the results,

Object-orientation -- to assign different attributes to an object and keep track of them throughout an object's life,

Dialogue boxes -- to specify a model's behavior,

Customization of blocks -- to modify and develop boxes with icons to help the user,

Hierarchical decomposition -- to decompose a box into its components for ease, and

On-line documentation -- to help the user with using different phases of the tool.

CHAPTER 2

METHOD DESCRIPTION

The first step was to choose SE principles which appeared to have relevance to BPR. Several standard software engineering textbooks were consulted (The New Software Engineering, Software Engineering: Methods, Management, and CASE Tools, and Software Engineering -- A Practitioner's Approach), and a list of applicable SE principles was constructed. This list was reduced by eliminating those principles which were more closely related to the technical aspects of software system production. The list, which applies more closely to the human activity aspects of software engineering, is shown in Table 1.

Software Engineering Principles	
No.	Title
1	Parallel Tasks
2	Interacting Tasks
3	Involvement of Workers & Management in Decision-Making Process
4	One General Version Process
5	Maximization of Reconciliation
6	Combining Several Jobs into One
7	Presence of a Project Manager

Table 1 List of Selected SE Principles

The second step consisted of identifying reviewing and defining the CASE tools which support the SE principles contained in the final list of Table 1. Again, several standard textbooks and other references were consulted to determine one or more CASE tools that would support each of the principles in the final list. Tools were classified according to function, technique, life cycle phase, and production process. After a detailed review of these classifications and definitions, the CASE tool components described in the work of QED Information Sciences, Inc. were chosen for use in this thesis.

Business Process Reengineering Principles	
No.	Title
1	Natural Order of Steps
2	Logical Place of Job
3	Decision-Making Process by Workers
4	Multiple-Version Processes
5	Minimization of Reconciliation
6	Combining Several Jobs into One
7	Presence of a Case Manager
8	Hybrid Centralization/Decentralization
9	Reduction of Checks & Controls

Table 2 A List of Selected BPR Principles

Next, the scope of BPR was delineated, and its principles were listed in Table 2. For this purpose, the work of Hammer and Champy (Reengineering the Corporation A Manifesto for Business Revolution) was selected. Their views with regard to BPR, as well

as with regard to the relationship between BPR and information technology, have been presented in several other books and journals, and are widely considered to be definitive. Consequently, they were selected to serve as the basis for the comparisons required for this thesis.

The final step was to examine the analogies between SE and BPR, and to determine whether these analogies were sufficiently precise to apply the techniques contained in the CASE tools for each SE principle to the analogous BPR principle. As indicated above, the final list of SE principles related to the human activity aspects of SE. Since the principles extracted from the BPR literature are almost exclusively related to human activity, the matching of the two groups of principles is based on the type of activity involved.

Chapter 3 discusses the results of the comparisons, and indicates the CASE tools that may be used to support each of the BPR principles.

CHAPTER 3

ANALYSIS: CASE TOOLS & BPR

Business Process Reengineering:

Definition:

As defined by Hammer and Champy (1993, p.32), "BPR is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service, and speed". From this definition, BPR sounds intuitive. But at times, it can be complex enough to introduce change into all parts of an organization. Any organization which wants to reengineer, in order to gain a new structure, has to give up its assumptions and conventional way of thinking and come up with new approaches toward getting the job done. "BPR rejects Adam Smith's industrial paradigm of achieving economies of scale through division of labor and hierarchical control" (Hammer and Champy, 1993, p.145). It works on combining several jobs into one with an adequate mix of centralization and decentralization. Most of other organizational procedures take deductive approach to organizational problems but BPR being an inductive as opposed to a deductive approach looks for the problems a given solution will solve instead of finding the solution to an existing problem.

BPR differs in several ways from Total Quality Management (TQM), Continuous Quality Improvement (CQI), automation, and reorganization. Even organizational

delaying, downsizing, or flattening can't be referred to as reengineering, as these describe some of the possible outcomes of BPR. It revolves around business processes, which are sets of activities creating valuable outputs for the customers. These activities are defined in simpler terms as "the way of doing work". On the other hand, reengineering is nothing else but "starting over". So, in informal terms, BPR put together can be thought of as "re-starting the way we perform work".

Features of Reengineered Business Processes:

Hammer and Champy (1993, p.50) said that reengineered business processes share some common features which are described below:

Combination of Several Jobs into One:

BPR emphasizes combining several jobs into one. Before the introduction of BPR, organizations all over the world used to believe in the integration of one process into several smaller processes. These smaller processes were performed by different workers, and so the outcome of one process was dependent on the performance of several specialized workers. For example, a job 'A' was subdivided into 'A1', 'A2', and 'A3' and then different groups of employees used to perform each of these different subdivisions. With the subdivision of each process, more misunderstandings and errors could be introduced. With many reengineered business processes, one individual handles the whole process from input to the output, thus becoming responsible for making sure that a particular customer's requirements are satisfied. The advantages of this feature include: improved control over processes, more

responsible employees, creation of innovative ideas to reduce process time, and introduction of less errors due to the reduction in the number of people handling a case.

Decision Making by Workers:

Individuals who are responsible for a certain outcome have full decision making power regarding that outcome. BPR questions the assumption that managers delegate authority and supervise workers. In order for workers to be held responsible for their work, it is important that they are given full authority for making decisions regarding the job for which they are responsible. Managers have been used to handling exceptional cases before reengineering, but with BPR the employees have full authority to handle exceptional cases along with the routine ones. This gives the managers a break from their routine jobs and lets them perform more productive work. In addition, it saves the overhead costs of appointing managers just to supervise employees and results in better customer response.

Natural Order of the Steps:

Steps in many business processes need not be done in the linear order which ties up a lot of time. Some steps don't need to be performed until some later stage in the process, and performing these steps earlier may waste a lot of useful time. In reengineered processes, employees start working on the next aspect of the job as soon as the prerequisite for that aspect is completed. They don't wait for finishing up the stages in a linear sequence. For instance, if they have acquired the information to start working on the fourth stage of the job before the third stage is finished, they will start working on the fourth stage instead of waiting for the third stage to be over. So, the different stages of a job start working as soon

as the initial stages for them have been accomplished. This strategy not only saves time but also reduces rework.

Multiple Version Processes:

Traditionally, most inputs have been handled identically, i.e., all of them went through the same process and same individuals to produce a uniform output. These traditional "one-size-fits-all" processes are quite complex. Most of the cases in such processes are handled through if-then statements i.e. if scenario one exists, then procedure one is chosen and if scenario two exists, then procedure two is chosen. Since specifications of the customers' requirements are changing in response to their changing needs, processes with multiple paths are becoming increasingly important. Each version of a process is handled differently according to the specifications of the case. Such versions are simple due to the absence of exceptional cases since each version is an exceptional case. This results in producing a product which satisfies customer needs and saves management time.

Logical Place of Job:

Companies and organizations which have not reengineered business processes may end up spending a large part of their resources on overhead costs, only because work is gathered around specialists instead of gathering specialists around the work. This gathering of work around specialists results in high overhead costs since work has to be carried at the workplace of specialists. Reengineered business processes relocates the work across organizational boundaries by reducing the work that individual organizational units perform. For instance, if the Information Systems Department could perform basic financial functions

without involving the purchasing and accounting department, it could save time and money for both the other departments. Business process reengineering reduces this part of the overhead cost and eliminates the integration of work by taking specialists to the work area.

Reduction of Checks & Controls:

As discussed previously, in business process reengineering workers make their own decisions. This reduces the need of supervising workers which in turn reduces the overhead cost of keeping a manager/supervisor just to look over employees' shoulder. Not that reengineered business processes do not have any sort of check, but the checks are aggregate or deferred. Organizations and management applying BPR do not believe in continuously monitoring the employees at every step they perform. The employees will have more time at their disposal to solve customer needs instead of watching their backs. These controls will either check the group output instead of an individual one or check at a later stage after allowing certain autonomy.

Minimization of Reconciliation:

Reengineering is all about getting the correct product or service to the customer in the shortest time. This goal can only be achieved if the organization works on first decreasing and then eventually eliminating the inconsistencies. Most inconsistencies are directly related to the number of external contacts a process has, therefore the smaller the number of contacts, the lesser the inconsistencies will be. A good example is a process where a case manager has to meet with customers several times during the process to find out their needs. If the customers will just provide the case managers with all the needs in the initial meeting,

the number of outside contacts will decrease for the managers, therefore; they will be able to perform their jobs more precisely. BPR involves making an effort to reduce the checking and reconciliation during the reengineered business process.

Presence of a Case Manager:

A case manager is a person who remains in direct contact with the customer throughout the whole process and is directly responsible for solving customer problems. Case managers need direct access to customer records through the information systems. Even though case managers are most useful while the processes are complex to understand or dispersed to integrate for the customer, they are usually working as "buffers" in all cases.

Hybrid Centralization/ Decentralization:

Centralization and decentralization each have advantages and disadvantages. Centralized operations tend to save money, while decentralized operations give autonomy to each department in an organization. Reengineered processes give autonomy of decentralization to each area of an organization. This technique saves resources through one major centralization of the organization with the help of information technology. Different departments can perform separate tasks independently but still be in complete touch with each other through updated softwares and computer systems in order to support each other's activities.

Software Engineering:

Definition:

SE is "an integrated set of methods, procedures, and tools for specifying, designing, developing and maintaining software" (Sommerville, 1992, p.56). All SE processes have some common features which are listed below:

Features of Software Engineering Processes:

Parallel Tasks:

Software processes, like business processes, involve parallel tasks. These tasks can be performed by different people as one individual can start working on the design while another one might still be getting specifications. One software engineer might not have to wait for the other one to finish taking the requirements from the customers. Next phase can start even while the previous phase is going on. This strategy can save up time and resources.

Interacting Tasks:

The individuals working on the processes (or on one process) have to continuously interact with each other due to the fact that one phase of the processes depends on the information gained in the other phase. This interaction between the tasks and the individuals designing the processes for the tasks during the whole SDLC is very important making troubleshooting a lot easier.

Involvement of Workers & Managers in Decision Making:

In a software development process, both the managers and the workers are usually involved in the decision making process. Managers make decisions about the resources and the budgets involved in the development. However, workers make decisions about the technical aspects of the software process, since they are in direct contact with the customers. In some organizations where the management is in direct contact with the customer instead of the software engineers themselves, managers are more involved in making decisions about the software implications.

One General-Version Process:

Even though a general purpose process engine is a solution in most of the software processes, software engineers must consider all cases that can take place during the process. A software engineered processes should be able to handle all versions of a particular task depending on the different inputs. A strong software engineered process is usually able to carry out different versions of the process to give outputs.

Maximization of Reconciliation:

Reconciliation is a major factor in software engineering since software engineers consult the customers usually during the whole SDLC. SDLC is an ongoing process to an extent. The information and the requirements about the process are not gained only at a time but they usually expand over the whole software development process. The reason for this is that the customers requirement sometimes change while the software is being developed and continuous update from the customers help software engineers design a better software.

Combination of Several Jobs into One:

Several jobs are combined into one because an individual involved in acquiring requirements and specifications from the customers is usually involved in designing the software too. Similarly a software engineer who has been involved in designing a software process can be involved in testing it since he/she has more knowledge of its weaknesses and strengths.

Presence of a Project Manager:

Software engineering processes are usually supervised by project managers. A project manager looks over the whole project since this individual is usually very well aware of the technical aspects of the process.

Applying CASE Tools & SE to BPR:

The reengineering gurus, Hammer and Champy (1993), said that many Information Systems (IS) people still too quickly confuse BPR with SE. These IS people think the issue is new systems for old processes. "It is not new systems for old processes. It is new systems and new processes" (p.76). While competitive pressures might be a powerful motivation for BPR, a major enabler is information technology. It is important to realize that while IT might not be necessary for BPR, it has been a critical component and facilitator of many successful BPR efforts. Reengineering is about eliminating work, and IS plays a catalytic role.

There is a style of working which seems to support both SE and BPR. SE and BPR came as solutions to software crises and organizational crises respectively. "Software Engineering is a disciplined approach consisting of a set of principles and goals that are applied during various phases of software development" (Sodhi, 1991, p.5). CASE tools support software design and engineering and helps software engineers by automating several parts of their jobs. BPR is an organization wide technique to improve business processes and tools like CASE tools can support BPR effort in an organization by automating several steps in BPR.

However, Hammer's position is that IS needs to be reengineered itself. He stated that "IT can not play any role other than that of an enabler in BPR" (Maglitta, 1994, p.85). Even though reengineering is very much integrated with the idea of process-orientation, Hammer did not see any connection between software engineering and BPR. He does not believe in SDLC and seemed to observe that BPR is not a one-time experience since companies will have to go through it on a repetitive basis.

The connection between different principles of SE and BPR is discussed below followed by a discussion of the application of CASE tools to various principles of BPR.

Combination of Several Jobs into One:

This feature of BPR emphasizes the whole process to be carried out by one individual instead of several specialized individuals. SE efforts are also concentrated on bringing in several different jobs together so that one individual can perform as many steps as possible. SE and BPR are both methods applied to get the output efficiently to the customer by

combining several different jobs into one so that one person is able to perform several steps. Several jobs can be combined into one through project management and can use all the CASE tools applicable in project management phase of SE.

Decision Making by Workers:

The decision making power of the workers usually leads them into project management. In BPR, emphasis is on the increase in decision making power of the workers where as in SE both management and workers together tend to make the important decisions. The management makes decisions about the finances and budgets of an SE process whereas workers are responsible for making decision during the technical phase of SE. Again, almost all CASE tools used during the project management phase of SE can be utilized for the decision making in BPR.

Natural Order of the Steps:

Steps during a BPR activity are not performed in a natural order. In fact, they are performed in any order which saves time and resources. This feature is analogous to the features of SE, where design analysis might start as soon as the requirements for a particular task have been collected. Not being restricted to following the natural order of steps introduces efficiency in the method. With the use of CASE tools which have high level data dictionaries, data bases, and logic analyzers, SE and BPR activities can be even more efficient. The presence of logic analyzer in a CASE tool which is to be used during a BPR process can help the individual reduce the number of logical mistakes and can help the steps

to be performed in the most efficient way. In addition, the data dictionaries and data bases can help restore all the information achieved at every step to be utilized at a later stage.

Multiple Version Processes:

SE usually designs out only one general version process but that general version process includes dealing with all different situations in a process whereas BPR emphasizes on having different multiple version processes. The CASE tools with strong design analyzers can help in designing different versions of a process in a BPR effort.

These CASE tools can help BPR management deal with every case on an individual basis.

Logical Place of Job:

This feature of BPR is not directly related to any principle of SE. However, the tools used during the unit testing phase of SE can be quite helpful in performing the job where it belongs. It can be supported through the use of context editors and language support software like compilers, linkers, and assemblers. The use of such CASE tools can help the job to be performed without much overhead costs and without the involvement of the organizational units that don't need to be involved.

Reduction of Checks & Controls:

In BPR, reduction of controls saves management time and gives autonomy and responsibility to the workers. This feature refers to decreasing the number of checks on individuals performing the process and does not comply with any certain principle of SE. Instead, SE believes in continuous checks of processes and implies nothing about checking the individuals. Process checking can be done during the unit testing and system testing

phases of SE and the CASE tools applicable in those phases can be utilized in BPR's phases.

Minimization of Reconciliation:

Minimization of reconciliation in BPR is getting most information about the requirements of the system from the user in the least number of meetings since more meetings with different people tends to increase the problems. System requirements and software requirements analysis phase in SE process correspond to this feature of BPR. SE principles emphasize on maximizing reconciliation and software engineers tend to go back and be in touch with their customers to get a constant update on their customers needs. The CASE tools helpful in these features of SE are prototyping tools, data dictionary, high-level data flow diagrams and object-oriented analyzers. These CASE tools help software engineers first gain and then record useful information from the end user of the process. Use of strong prototyping tools and object-oriented analyzers can help minimize reconciliation during a BPR effort.

Presence of a Case Manager:

The case manager, in BPR, is the individual responsible for meeting with the customer and figuring out the needs and outcomes of the process. This individual solves customer problems at all phases instead of different people solving different problems of the customers. Similarly, project manager in SE oversees all the phases of SE. CASE tools used by project manager can also be utilized by the case manager to keep the customer aware of the current stage of the process and to deal with the questions customers might have about the project.

Hybrid Centralization/ Decentralization:

The organizations exercising BPR emphasize on a mix of centralization with decentralization. Again, all phases and principles of SE can be encompassed indirectly by this feature of BPR. However, none of the principles of SE directly complies with this principle of BPR.

CHAPTER 4

CONCLUSION & RECOMMENDATIONS

The era of the 1990s has been characterized by slow growth and high competition; product leadership and customer intimacy separate the winners from the losers. This competition has led organizations to perform their processes faster by splitting bigger jobs into smaller jobs to be performed by more specialized individuals. But after years of splitting jobs into smaller pieces and creating inward-focused, highly matrixed organizations, executives at many large organizations are realizing that something is very wrong. The functional divisions are intently focused on their own small piece of the pie rather than on the big picture: creating great products and satisfying customers. Hammer's and Champy's response to the problem is to start from scratch, look at the end-to-end processes that are really important to a company's success, then rapidly redesign job processes and give workers new tools with which to get more done without being limited to current organizational structure or current thinking.

This concept called 'BPR' is now becoming an increasingly crucial issue facing most of the organizations all over the world. "BPR is process-oriented; it concurrently pursues breakthrough improvements in quality, speed, and cost; it is holistic, both leveraging technology and empowering people; and it starts with a willingness to abandon current practices" (Klein, 1994, p.30).

While BPR is an approach to reengineering business processes, SE is an approach to engineering software system. The purpose of both SE and BPR, however, is to meet user requirements and expectations within available resources.

Since SE and BPR both are processes, computerized tools can help perform these processes in a more efficient way. Tools like CASE that are being used now for SE can be utilized by BPR. Analysis of different principles of BPR reveal the presence of certain features which are analogous to SE and can be performed using CASE tools. CASE tools can save time and resources, and can help BPR efforts in several different ways.

The research reported herein has shown that SE and BPR have several principles in common. These principles include Combining several jobs into one, Presence of a Case/Project Manager, and Not following natural order of steps. Besides these principles, BPR includes some other principles which can be partially supported through CASE tools. Such principles are: Decision making by Workers, Multiple version processes, Performing the job at its logical place, Reducing checks & controls, Hybrid Centralization/Decentralization, and Minimizing reconciliation. The CASE tools that support these BPR principles are:

- ▶ Project Management Tools for Combining several jobs into one
- ▶ Project Management Tools for Not following natural order of steps
- ▶ Design Analyzer Tools for Decision making by workers
- ▶ Context Editors & Language Support Software Tools for Performing the job at its logical place.
- ▶ Unit Testing Tools for Reducing checks & controls

- ▶ Prototyping Tools & Data Dictionary for Minimizing reconciliation
- ▶ Project Management Tools for the Case manager

CASE tools can help BPR managers carry out detailed design analysis through data flow diagrams. Managers can sketch the whole system to be reengineered and the new reengineered system with the help of these data flow diagrams. Language support softwares can help them choose the appropriate languages to be used during BPR activities and overcome the language deficiencies. Project management tools can help BPR professionals in each and every step of BPR activities.

The SE principles have led to specific procedures through which software engineering can be implemented in certain situations. The next step in BPR research should be to come up with procedures to help implement BPR in an organization. Hammer and Champy who initially coined the term BPR and have discussed BPR in detail but have not mentioned much about the specific procedures through which managers can implement BPR in specific situations. Further research needs to be carried out not only to come up with such procedures through which managers will be able to operationalize BPR in different situations. Also, there is a need for further research to be done in order to come up with the metrics through which the extent of a BPR effort's success can be measured.

REFERENCES

- Cavanaugh, H. A. (1994, April). Reengineering: Buzz word, or powerful new business tool? Electrical World, 7-15.
- Chikofsky, E.J. & Rubenstein, B.L. (1988, March). CASE: Reliability Engineering for Information Systems. IEEE Software, 11-16.
- Conger, S. (1994). The New Software Engineering. California, Belmont: Wadsworth Publishing Company.
- Flecher, T. & Hunt, J. (1993). Software Engineering and CASE: Bridging the Gap. New York, New York: McGraw-Hill, Inc.
- Fuggetta, A. (1993, December). A Classification of CASE Technology. IEEE Computer, 26, 25-38.
- Hammer, M. & Champy, J. (1993). Reengineering the Corporation A Manifesto for Business Revolution. New York, New York: Harper Collins.
- Hansen, G. A. (1994, September). Tools for Business Process Reengineering. IEEE Software, 11, 131-3.
- Klein, Mark M. (1994, Spring). Reengineering Methodologies and Tools: A Prescription for Enhancing Success. Information Systems Management, 30-35.
- Maglitta, J. (1994, January 24). One on One with Michael Hammer. Computerworld, 84-86.
- Martin, E. W., DeHayes, D.W. & Hoffer, J. A. (Eds.). (1994). Managing Information Technology. New York, Macmillan.

Moad, J. (1993, August 1). Does Reengineering Really Work? Datamation, 22-28.

Parker, Jon. (1993, May). An ABC Guide To Business Process Reengineering. Industrial Engineering, 52-3.

Pressman, R.S. (1992). Software Eng. -- A Practitioner's Approach. New York: McGraw-Hill.

QED Information Sciences, Inc. (1989). CASE The Potential and the Pitfalls. Massachusetts: QED Information Sciences, Inc.

Huff, K. E. (1993). From Software Process Engineering to Business Process Reengineering. In W. Schafer, (Eds.), Proceedings of the 8th International Software Process Workshop (pp. 98-101). Los Alamitos, California: IEEE Computer Society Press.

Sodhi, J. (1991). Software Engineering: Methods, Management, and CASE Tools. Pennsylvania: McGraw-Hill.

Sommerville, E. (1992). Software Engineering. Massachusetts: Addison-Wesley.