IMPEDANCES TO RE-ENGINEERING - A DIFFERENT PERSPECTIVE

by

Jean K. T. Low

B.Sc., University of British Columbia, 1980

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the School

of

Computing Science

SIMON FRASER UNIVERSITY

August 1993

Approval

Name: Jean K. T. Low

Degree: Master of Science

Title of thesis: Impedances to Re-Engineering - A Different Perspective

Examining Committee:

Chair: Dr. Dave Fracchia

Dr. Nick Cercone
Senior Supervisor
Professor of Computing Science

Dr. Drew Parker
Associate Professor of Business Administration

Dr. Christopher Jones
External Examiner
Associate Professor of Business Administration

Date Approved: 30 August '93

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

Impendances to Re-Engineering - A Different Perspective.

Author: _____
(signature)

Jean Kit Tsang Low
(name)

Sept 7, 1993
(date)

Abstract

Business Process Redesign (BPR) is emerging as a competitive imperative for the 1990s. It is a revolutionary business concept to effect monumental productivity gain in an organization. BPR usually entails either or both of software and hardware re-engineering. However, the essence of the true BPR concept is seldom well understood, resulting in ultimate systems failure. Many seemingly minor, unrelated, or often overlooked issues contribute to such failure. This paper examines a selected few and explores how such issues impede systems re-engineering efforts.

"There is nothing more difficult to take in hand, more perilous to conduct or more uncertain in its success than to take the lead in the introduction of a new order of things."

Niccolo Machiavelli, The Prince (1532)

Acknowledgements

First and foremost, I would like to express my gratitude to my senior supervisor, Dr. Nick Cercone, for turning my problems into an opportunity, the impetus for my thesis topic. I would also like to thank Dr. Drew Parker for his valuable suggestions for improvement; Dr. Wo-Shun Luk for his challenges and critiques; Dr. Chris Jones, for his gracious acceptance as my external examiner; Dr. Lou Hafer, our graduate director, for his support in pushing through the most difficult last mile. And last but not least, my appreciation to Kersti, for being such a sweet grad secretary; Joseph, for his unwavering long-distant moral support.

Table of Contents

1. Introduction

1.1 Background

In the late 1970s, with books like "*In Search of Excellence*" and "*Theory Z*" amongst the best sellers, American corporations began to concede to the business inadequacies of their management approaches. Many companies revamped their corporate cultures and introduced "excellent" company's or Japanese-like business concepts such as Quality Circles, Synergy, MBWA (Management By Walking Around) etc. into the work place, to booster morales, with the expectation of improved productivity.

While the implementation of these concepts may improve productivity, they are nevertheless improvements of the *micro* scale. As contemporary technology pushes away transportation and communication barriers, shrinking global information float[1], trade and commerce not only must compete domestically, they must compete globally. International trade necessitates a whole new business and financial mindset. In order to compete globally, a business must improve productivity on a *macro* scale. Little micro improvements, no matter how numerous, are no longer adequate.

Modern businesses desperately need some innovative niche to survive global competitions. As if in answer to their prayers, a brand new revolutionary business model emerged in the mid 1980s. This miracle quality breakthrough is called ***Business Process Redesign*** **(BPR)**. It amounts to no less than re-examining all of the corporate operations and re-structuring the whole organization to meet its business objectives. Thus another term commonly associated with BPR is "*re-engineering*".

It is no wonder that the wave of new quality concepts: BPR, Object Oriented (OO) Paradigm, Software Engineering, Expert Systems (ES), all have a similar touch and feel to them, since these concepts, besides having many common elements, overlap, mirror, enhance, cross-reference and reinforce each other. It is the advent of many of these new technology and concepts which makes BPR possible. However, the age old problems of project management persist as impedance to re-engineering.

---

[1] As data transmission from different corners of the world becomes faster and thereby more frequent, information floats more freely and more liberally around the globe.

There are publications on how to be a good manager. There are publications on how to be a good programmer. There are even some publications on how to manage the special breed of technical staff. But one is hard pressed to find a detail account of the subtleties a manager must appreciate in order to motivate his IS (Information Services) staff to achieve the best efficiency. This paper fills in some of these missing gaps by documenting these subtleties which affect IS productivity, from the perspective of experienced IS professionals, the grassroots. This paper claims that these subtle issues contribute to an re-engineering project's failure. These failing factors not only interfere with a project's smooth operations, they also raise its hidden costs, a very important corporate concern. By highlighting the implications and ramifications of these interfering factors, this paper should serve as an impetus for management to minimize the impacts of such problems, many of which are inherent parts of corporate reality, thus impossible to eradicate.

1.2 Outline of Dissertation

Chapter 2 outlines the evolvement from the traditional approaches of organizational management to the current concepts of Business Process Re-engineering (BPR). Chapter 3 defines the true meaning of BPRs, dwelling briefly into the important roles played by Information Technology, Expert Systems and Object Oriented Paradigm. Chapter 4 focuses attention on systems re-engineering in particular, narrowing the scope of analysis. Chapter 5 discusses the motivation and hindrances to re-engineering, as occurred at different stages of a project's life cycle, singling out the chief impedance in the Decision Phase. Chapter 6 follows through with the argument presented in Chapter 5 and elaborates on how cost factors affect a project decision. Chapter 7 details the various issues which can impact costs and adversely affect a re-engineering project. Chapter 8 examines a case study which involves most of the issues discussed in the preceding chapters. Chapter 9 concludes the paper by summarizing how costs and people issues are interrelated and how implicating one issue also implicates the other.

2. Traditional Approaches to Organizational Management

2.1 Conventional Wisdom

Almost all business processes predate the advent of modern computer and communication technology, even as far back as the Industrial Revolution. *Specialization of labour* and *economies of scale* were introduced to overcome the inefficiencies of cottage industries. Sometimes, *division of labour*[1] was established to provide checks and balances in an organization.

For instance, the Payables Department, which holds the exclusive right to remit all payments, is physically separated from the Purchasing and Warehouse Receiving Departments. Control is enforced via a paper audit trail, which imposes that invoices be matched against authorized purchase orders (PO) and validated packing slips. Today, data are entered directly into the computer. But instead of electronic verification, POs are still printed and manually matched against invoices, defeating the goal of computerization to achieve a paperless office.

2.2 Taylorism

Taylorism has prevailed in work organizational management since the turn of the century. Frederick Taylor revolutionized the workplace with what we now call *Industrial Engineering* (IE) - the concept of increasing organization productivity by applying engineering principles, such as task decomposition and job measurement, to human labour. [DAV90]

Today's business procedures, as conceived in Taylorism, are usually organized as a sequence of separate tasks, albeit automated, handed off from person to person or unit to unit, making delays and errors inevitable. Such fragmented and piecemeal approach is the result of following rules of work design left over from earlier decades, rules which are based on assumptions about technology, people, and organizational goals that no longer hold. [TER91]

---

[1] By dividing a task into smaller modular units, even unskilled labour can be trained readily to perform efficiently (*Division of Labour*). Each worker can become proficient at his unit of work (*Specialization of Labour*), thus producing more and at a faster rate. The more goods are produced, to share in the fixed cost, the lower become their unit costs, thus achieving *Economy of Scale*.

2.3 Why are Inefficient Processes Designed in the First Place?

Many business procedures were not designed at all - they just happened to be improvisations to solve upcoming problems and staff adjusted its work accordingly. Once installed, this "hodge-podge of special cases and quick fixes was passed from one generation of workers to the next".

Over time, the *ad hoc* become institutionalized and the *temporary* become enshrined. "Why does an electronics company spend $10 million a year to manage a field inventory worth $20 million? Once upon a time, the inventory was worth $200 million, and managing it cost $5 million. Since then, warehousing costs have escalated, components have become less expensive, and better forecasting techniques have minimized units in inventory. But the inventory procedures, alas, are the same as always." [HAM90]

2.4 The Change of Time

Traditional processes are replete with mechanisms designed to compensate for *information poverty*[1]. Over time, these mechanisms become enshrined in automated systems. Although businesses nowadays are *information affluent*[2], they still use the same outdated mechanisms, without questioning "why?" or "what if?". [HAM90]

Taylorism is effective on workplace rationalization and individual task efficiency for a stable business environment, a luxury of the past era. However, as corporations enter a new era of rapid changes, this outmoded IE is no longer adequate. So, instead of simply focusing on *individual* tasks, companies nowadays must develop more flexible, team-oriented, *coordinative* and communication based work activities *across* the entire organization. [DAV90]

---

[1] In the olden days, most of the data was exchanged via the paper medium. Therefore the amount of information obtained or transmitted was thus limited by such factors as the speed and accuracy of the typist, postal delivery, ground transportation and so forth.

A simple report might take weeks to compile since at office X, the figures had to be gathered by clerk A and then copied by typist B, to be mailed to office Y, where statistician C performed the analysis to be transcribed by typist D, so that the finished report could be eventually mailed back to office X.

[2] With the latest technology in data communications, such as electronic mail, bulletin boards and data interchange, information can be obtained or transmitted almost instantaneously.

The same report mentioned earlier can be displayed on the screen as soon as the data is fed into the computer at source. Moreover, the electronic report is available to users world-wide, to be used for different purposes, such as spreadsheet computation, graphical presentation, decision analysis, etc.

3. BPR (Business Process Redesign)

A recent CSC Index Inc. (a Cambridge, Mass. consultancy) report indicated that nearly three-quarters of the 407 large U.S. and European firms surveyed now have a "major formal process improvement effort" under way and the high-ranking IS executives of these businesses ranked "re-engineering business process" high on the list of the ten most pressing challenges for 1993. [MAR93a]

3.1 Definition

A Business Process is a set of logically related tasks performed to achieve a defined business outcome. Business processes have two important characteristics: they have customers (internal and external), and they cross functional and organizational boundaries. [DAV90]

For all intents and purposes, redesign is synonymous with re-engineering. As offered by Michael Hammer and James Champy, two leading consultants in the field of business re-engineering, BPR is "the fundamental rethinking and radical redesign of an entire 'business system' - the business processes, jobs, organizational structure, management systems, values and beliefs to achieve dramatic improvements in critical measures and performance.

"... Business re-engineering is discarding conventional ways of working and replacing them with entirely new ones. ... re-engineering is about massive, multidimensional, holistic business change." [HAM92]

"BPR is not a new concept and can be implemented using information engineering and structured analysis and design techniques. BPR is based upon the concept that enterprises are systems (processors) that transform capital, labor and raw materials (inputs) into goods and services (outputs). Thus, organizations can be described, analyzed and defined using entity-relationship, data-flow and related system modeling techniques. Like an information system, the enterprise should exhibit loose coupling (i.e., organizational units should be able to work assynchronously, passing data and/or materials across organizational boundaries) and should also exhibit high cohesion (organizations and enterprises should be structured so that the work performed by, and deliverables produced by, each organizational unit are closely related)." [CAS91]

3.2 The Essence of BPR

The key to BPR is *restructuring*, not automation. BPR challenges old assumptions, restructures antiquated business processes, replacing underperforming culprits, to achieve breakthroughs in performance. BPR efforts are characterized by radical changes, achieving quantum leaps in performance, by focusing on eliminating waste and bureaucracy.

To succeed, BPR must be given strong *executive leadership* and be managed as a *business* and *operations* endeavor, **not** as a *technology* endeavor. [HAM92, TER91] "It requires a lot of thought, a well-developed plan, a sophisticated approach, and unfaltering leadership." [HAR90, CAF93]

In order to achieve the *integration* required to maintain quality and service, organizations must recognize and break away from outmoded rules and obsolete fundamental assumptions that underlie operations. They should "solve problems that never should have occurred in the first place." [HAR90] A re-engineering team must keep on asking "*Why*?" and "*What if*?".

BPR negates virtually all of the premises on which work design and organizational structures have been based for the past century and a half. Instead of work fragmentation and task specialization, BPR focuses on task compression and integration. Instead of linear and sequential process structures, BPR proposes parallel process structures. Instead of hierarchies for decision making, BPR demands universal sharing of decisions. Instead of trade-offs between centralization and decentralization, BPR permits a hybrid of the two.

BPR should be a *dynamic* and *continuous* process, which is always monitoring the altered business processes to determine if additional, future alterations are required. As soon as one business function is addressed, we should strive to identify another critical-path business function as the next candidate. [CAS92]

The three mottos to re-engineering are: [SCH93]

(1) Think before you automate.

(2) Redesign before you initiate.

(3) Fix before you integrate.

3.3 Information Technology (IT) and BPR

Just as Taylorism has transformed organizations since the turn of the century, two contemporary tools are claiming the same credit in the 1990s - *Information Technology* (IT) and *BPR*. These two tools compliment each other, having the potential to create a new type of IE, changing the way the discipline is practised and the skills necessary to practise it.

IT plays a critical role in BPR. It is fundamental to re-engineering, being a key enabler in virtually all such initiatives. Practically every company that has re-engineered its business processes has required IT support in the form of technologies, tools, infrastructure and people.

Until the emergences of the latest enabling technologies, such as object orientation (OO), client/server, knowledge-based systems (KBS), graphical user interfaces (GUI), end-user computing (EUC), desktop workstations, open systems, Unix and so on[1], traditional development and delivery technologies rarely have had the sophistication or the horsepower to make many BPR applications viable. [BRA91a]

Without IT, it would be impossible to depart from some of the old rules and conventions which dictated the past design. For example, without Electronic Data

---

[1] *OO* is a modularity modelling concept whereby each entity can be represented as an object of a particular class(ification). Classes are modelled within a hierarchical classification where subclasses (children) inherit properties from their superclass (parent).

Client/server is a network technology whereby a specialized *server* (e.g. a DBMS) at one node services all the requests from other *client* sites within the network.

A *KBS* is a database system containing rules as data. The rules represent knowledge, usually used by an expert system.

GUI is another user-friendly operating environment in which a user can access software applications via graphical icons, instead of by issuing text commands. The systems in question usually involve multiple applications which can communicate with each other through graphical interfaces.

EUC is made possible with the new generation of user-friendly menu- or window-driven operating system shells and other software facilities. Not only can layman users readily access a wide spectrum of computerized business application softwares, such as spreadsheets, databases, financial analysis packages, etc., they can also easily integrate multiple functions, passing data from one application to another. No longer are they dependent on programmers for assistance, even for sophisticated outputs involving texts, graphics and pictures combined.

Open Systems are the new generation of computing standards adopted by all participating vendors. The standards provide a means whereby data and softwares generated by different vendors can communicate or interface with each other, even across hardware platforms.

Transfer, it would be impossible to implement a paperless invoice system between suppliers and purchasers.

Expertise with IT is essential for creative redesign. For example, it would be impossible to conceive a solution to control a transportation network without a thorough knowledge of the availability and features of the most up-to-date technology such as image processing, desktop workstations, optical disk, and networked file services.

3.4 Analogy of IT Investments to Stock Options

IT infrastructure investments is somewhat analogous to stock market options, where a *call* option allows an investor to buy and a *put* option allows him to sell a stock at a predetermined price at a future date. With a proper IT infrastructure in place, management has the ability to readily build new applications impossible without such infrastructure, analogous to exercising a *call* option, in the sense that the cost is scoped within a certain range. Similarly, with such infrastructure, management has the flexibility to quickly replace existing applications no longer meeting the needs, analogous to a *put* option, in the sense that the loss is limited to an acceptable level. For instance, with a distributed system in place, a company can easily install electronic mail (*call* option)[1]. Similarly, investments in standard-based equipments allows a company to abandon software from a particular vendor without worrying about also replacing the hardware as well (*put* option). [CAR92]

3.5 Misconceptions about Re-engineering

According to Hammer and Champy, re-engineering "is not automating or re-automating existing business procedures, the paving and repaving of the cowpaths that have characterized the computerization of businesses for the last 40 years. This is a common mistake made by some IS advocates of information engineering and CASE methods." [HAM92]

BPR is not an IS function, but rather a business operations or management function. Many systems professionals confuse business re-engineering with software re-engineering, misinterpreting re-engineering as simply re-implementing old systems with new technology, applying IT to mechanize old ways of doing business, leaving the existing

[1] The analogy here is the exact reverse of what has been given in the referenced article.

processes intact, using computers simply to hasten office work, missing the point altogether. [CAS92]

BPR is more than simply rearranging and automating business processes, it is a total *transformation*. Automating a bad process only ensures that the badness is repeated every time, faster, and with less effort. Thus, unless we change the outdated rules which conceived those bad processes in the first place, "we are merely rearranging the deck chairs on the Titanic". [HAM90]

3.6 Some Sample Cases on Re-engineering

3.6.1 Equitable Resources, Inc. (ERI)

Some companies re-engineer to realize substantial cost savings. Equitable Resources, Inc. (ERI), migrated its IBM System/36 to a distributed system based on PCs and laptops. It "calculated that a million instructions per second (MIPS) on the desktop costs $100 while a MIPS on the mainframe goes for $100,000." [SCH92]

This is a typical example of hardware downsizing, which is becoming increasingly widespread and popular, as micro power per cost ratio far exceeds that on the mainframe.

3.6.2 Mutual Benefit Life (MBL)

To improve customer service, Mutual Benefit Life (MBL) dispensed with existing job definitions and departmental boundaries to create a new position called a *Case Manager*, who would take on the total responsibility for all tasks dealing with insurance application, all the way through to issuing policies, supported by new technology tools such as work flow systems and expert systems.

Handling insurance applications at MBL used to be a long, multi-step process going through "as many as 30 discrete steps, spanning 5 departments and involving 19 people". Expert System *expedites* a smooth transition and enables MBL to consolidate all the tasks of credit checking, quoting, rating, underwriting, etc., into a unified, cohesive function, easily managed by a single autonomous *Case Manager*.

The "case managers can handle more than twice the volume of new applications the company previously could process." And with the new operations, the average turnaround takes 2 to 5 days, instead of 5 to 25 days as required previously. [HAM90]

3.6.3 Ford Motor Company

Most companies re-engineer under extreme circumstances - such as when their competitors are many times more efficient than they are. In the early 1980s, when Ford discovered that it had 500 accounts payable clerks while Mazda had just 5, it finally instituted an "invoiceless processing" procurement process with the aid of EDI (Electronic Data Interchange). [SCH93, HAM90, FLY92]

Instead of each clerk handling various functions of processing the purchase orders, invoices, receipts, payments, and so forth, passing paper data from one department to the next, for verifications and authorizations, etc., at each stage, matching one form against another, the new BPR bypasses the intermediate steps and eliminates all the paper procedures. Purchase orders and invoices between Ford and its suppliers are now transmitted via EDI. By replacing the old rule of "we pay when we get the invoice" with the new rule "we pay when we get the goods", Ford has achieved a 75% reduction in head count.[1]

3.7 The Role of Expert Systems

Expert Systems (ES) goes far beyond simple automation, which amounts to no more than using the computer to speed up a work function. ES draws on the specialized skills and knowledge of experts to derive an *expedited* decision choice or solution alternatives. ES is especially valuable in *time-critical* applications such as process control, program trading, threat assessment, network management, and so on. [DAY88]

As ESs become more commercially prevalent, more and more enterprises are beginning to turn to ES to improve customer services and goodwill, perceiving it as a means to speed up customer responses and to avoid costly human errors.

3.7.1 Real-Life Example

Many BPRs incorporate ES to enhance productivity. As in the case of Mutual Benefit Life (see Section 3.6.2 on Sample Cases) where a single new position of *Case Manager*, aided by an ES, has replaced 100 field office positions.

---

[1] No mention of drain on cash flow or carrying cost type issues with Ford's new procurement procedures.

Tougher economic conditions force even once-complacent companies to leap to re-engineering. Pacific Bell's effort is such an example: it resorted to expert system to recoup more than $1.5 million of unbillable calls each month. In 1992, it cut 4,300 jobs and is expected to trim 11,000 positions by 1994. [SCH92]

3.7.2 Catalyst for Change

With Expert Systems (ES) providing interactive support and guidance, results-oriented generalists can accomplish the jobs of task-oriented specialists. By expediting the transition process, ES is a *catalyst* for change, a valuable contributor to re-engineering.

3.8 Parallelism to Object-Oriented Paradigm

Object technology fits nicely into BPR. Each business process can be readily viewed as a complex *object*. Organizational hierarchies are easily represented by *class hierarchies*. Multidimensional processes cutting across organizational boundaries can be modelled with *multiple inheritance* and *polymorphism*[1].

As mentioned in Section 3.2, the shift from fragmentation to integration, from sequential to parallel structures, from hierarchies to universality, can only be readily accomplished through the flexibility and versatility offered by OO.

Natasha Krol, an industry analyst with META Group, Inc., stated, "The process of building class hierarchies forces developers to address essential business principles as they create the basis for subsequent applications by building the foundation business classes. Object technology is a better fit with business process re-engineering than are information engineering and present CASE tools. These tools do not motivate analysts to focus on the origins of the business problems themselves. Consequently, approaches like information engineering tend to mechanize or legislate the old ways of conducting business."

Besides putting objects to work in corporate downsizing and re-engineering efforts, some sites are even turning to object-oriented "wrappers" to hide procedural code by encapsulating them as objects. [BOZ92a]

---

[1] *Polymorphism* is the capability of different classes of objects to respond to the same message (procedure call or command), or alternatively, the capability of different classes to define methods (procedures) with the same name. E.g. Regardless whether it is a LaserJet or dot-matrix, each different printer object has a generic method called "print".

Many expert system concepts can also be suitable participants in OO data models. For example, each of the *event, condition* and *action* entities in an ECA rule can be viewed as distinct objects. Also, much of the GUI-related (Graphical User Interface) technology is based on OO designs. [DAY88]

3.8.1 Real-Life Example

A pseudo OO example may be observed in B. C. Telephone Company's CRIS (Customer Record Information System) system transaction practice.

IMS/DC[1] maintains a conversational (interactive dialogue) session for database access by passing messages (formatted screen data) between the user terminals and the application programs (transactions). Any data necessary to maintain a dialogue (continuous stream of request-responses) is normally stored in a SPA (Scratch Pad Area).

For efficiency consideration, CRIS bypasses the SPA's and makes each transaction self-contained by storing the necessary conversational data within the hidden fields of each message itself. This innovative design concept has made it possible to conceptualize such pseudo-conversational transactions as individual *objects* and thus can be readily fitted into the Object Oriented paradigm of any new system redesign.

3.8.2 Reusability

OO and re-engineering shares a common goal - *reusability*. Reusability incurs up-front investments and demands skills: skill in terms of programmers who originally implement the reusable codes,[2] skill in programmers who subsequently reuse these codes,[3] and skill in the systems librarians who must recognize when and how these codes are to be reused.[4] [RAY92, McC92] (Refer to Appendix)

Developing reusable components is generally more expensive than developing specialized code. [CAL91] According to W. Tracz of Stanford University, "code generated

---

[1] IMS is a hierarchical Data Base Management System developed by IBM in the 1960s. IMS/DC (Data Communication) is the online system for IMS.

[2] (e.g. component semantics and interface; composition, systems, and domains)

[3] (e.g. component customization and algorithm reuse)

[4] (e.g. algorithm cataloging, conceptual distance graph, dissimilarity coefficient, subsumption relation, Domain Analysis etc.)

for reuse (reuseful code) might cost 30 to 200% more to develop, document, and test, but subsequent reuse costs 20 to 40% less than rewriting". [TRA88] Lanergan and Grasso found rates of reuse of about 60% in business applications. [CAL91] However, with a strong management mandate, investments in reusability can be economically viable. For example, GTE began software reuse engineering in 1986, "when a vice president heard about software reuse at a conference and then started a small program. They began by collecting common program utilities that were called *assets* ... GTE reports on average a 20 to 30 percent increase in productivity when a new development or major enhancement project uses software reuse. ... GTE reports almost zero defects found in reusable components ..." "In Japanese software factories, an 85 percent software reuse level helped increase software productivity eightfold." [McC92 P.221,222,227]

"Both the Japanese[1] and the Raytheon[2] experiences point out that for reuse to be successful it must be practiced at the organization level, not simply at the individual level. ... usually, a minimum of (being reused) three times is required to reach a payback point. Practicing reuse at the organization level requires the development and acceptance of organization standards." [LAN, McC92 P.230]

Current researches tapping into reusability (such as AIRS [OST92], OBSERV [TYS92] Genesis, Avoca, [BAT92][3] PA and CARE [McC92 P.243][4]) have the potential to

---

[1] The Japanese are the world masters of software reuse ... also the world masters of software standardization, which is the basis for their success with reusability. [McC92 P.229]

[2] Raytheon began its search for common software components in 1976 when the company realized that many software functions were common across its COBOL application systems. [LAN, McC92 P.229]

[3] AIRS is an AI-based library systems for software reuse. ... One important aspect of AIRS, ..., is the ability to reason heuristically about the similarities between desired components and components residing in the existing knowledge-base (software library). [OST92 P.210]

In the OBSERV methodology, the designer defines a system by identifying objects and the relations between them. The methodology makes a distinction between these two aspects to enhance the possibility of reuse. The OBSERV language, to a great extent, isolates each object's definition from the way the object is used within a broader system. [TYS92 P.274]

The Genesis 2.0 (G2) prototype implements a portion of a domain model for database management systems. G2 enables centralized, single-client DBMSs to be synthesized from component libraries. [BAT92 P.361]

Avoca is a system for constructing efficient and modular network software suites using a combination of preexisting and newly created communication protocols [BAT92 P.356]

[4] PA (Programmer's Apprentice) is currently under development at the MIT Artificial Intelligence Laboratory. It is a program development system whose ultimate purpose is to automate the programming process. [McC92 P.243]

reduce the cost of software development. "MCC has built an experimental design reuse system called ROSE-2 that supports the *whole approach*, ... an analysis method that starts with a reusable program design architecture and modifies it in steps to meet the requirements of the new program." [McC92 P.260,261] Reusability is a topic of its own right and beyond the scope of our current discussion. (Refer to Appendix)

3.8.3 Parallelism

Viewing business processes as objects facilitates BPR as each object naturally represents a real life model of the actual process. No longer do developers have to juggle their designs to fit the requirements into the constraints dictated by the tools, be they programming languages, databases or data communication softwares.

OO design encapsulates the complexity and hides the low level details, thus enhances modularity and promotes ease of conceptualization. Encapsulation permits both users and implementors to view the same object model from their own different perspectives, enhancing mutual understanding and thus reducing communication breakdowns.

CARE (Computer-Aided Reuse Engineering) is developed by the Computer Science Center at the University of Maryland to extract and measure reusable components from ANSI C and Ada programs. [CAL91, McC92 P.251]

4. Systems Re-engineering

4.1 Three Forms of Re-engineering

The term *re-engineering* has been proliferated to many areas. It may refer to the restructuring of a business process (BPR), of a software systems application, or of a hardware platform. An example of software re-engineering may be to convert from hierarchical to relational or OO databases. An example of hardware re-engineering may be to convert from mainframe to client/server technology.

One form of re-engineering tends to necessitate and to entail another. For instance, most BPRs are achieved via introduction of new information technology (IT), implying either or both of software and hardware re-engineerings. Similarly, a conversion from one hardware platform to another is usually accompanied by a change in operating system softwares.

4.2 Some Statistics on Re-engineering Projects

The average company has more than doubled its number of re-engineering projects in 1992, an 175% boost from 1991. However, as Jeff Plewa, a partner at Deloitte & Touche commented: "I think there is a tendency to call any sort of restructuring or cost-cutting effort re-engineering." [CAF93]

Recent polls showed that more than 70% of large U.S. companies claim to be re-engineering. According to a ComputerWorld survey, one in four firms involved in re-engineering projects is considering total, large-scale redesign. Six companies are venturing into megaprojects with budgets over $100 million. However, most firms are doing pilot projects involving less than 30 people and under $1 million in funding. [CAF93]

CSC Index, Inc. reported that nearly three-quarters of re-engineering projects would fall short of expectations. According to CSC and Hammer, almost one-quarter of the 300 re-engineering projects in North America were not meeting their goals. The authors speculated failure rate was more likely "on the order of 70%". Some feared that the number was even higher. [CAF93]

4.3 Systems Re-engineering in Particular

In the 1990s, software re-engineering has gained popularity as companies try to control their software maintenance costs and maximize their investment in existing software. In particular, those companies with a large portfolio of aging applications that need upgrading are looking to recoup their investment in existing software process logic.

"Re-engineering helps an organization move away from reactive maintenance to active management of its production system portfolio. ... Re-engineering is the positioning technology. Re-engineering can upgrade existing systems to the latest technologies and position them to take advantage of integrated CASE environments. Also, re-engineering can help populate repositories[1] with enterprise and software system information." [McC92 P.23,267]

Whereas some of the most outstanding Business Process Re-engineering endeavours involve relatively little cost outlay to reap large cost savings, (for instance, Ford's Payable streamlining where the solution amounts to restructuring some existing operations) a systems re-engineering project tends to be very labour intensive. Thus it can only be justified if it has a very rewarding cost/benefit ratio and relatively short payback period.

Systems re-engineering is a three step process: *reverse engineering*, *specification model revision* and *forward engineering*.

4.3.1 Reverse Engineering

Reverse engineering is the reversal of system processes with structured techniques. Its primary function is to capture functional capability or process logic of the existing system in a simplified and structured form.

There are two major reasons for reverse engineering: to save on software maintenance cost; and to clean up legacy systems in preparation for moving applications to different hardware platforms. Since on average, 60% of the time spent on maintenance is related to figuring out what the system is doing, reverse engineering should cut such cost by 25% to 30%. [NAS92, RAY92b] Reverse engineering and reusability are closely

[1] See subsequent section (4.3.4) on Re-Engineering Methodology and Tools.

linked. Systems such as Rigi under research at University of Victoria are working towards such goals. [MUL, TIL92a, TIL92b]

4.3.2 *Specification Model Revision*

A system which warrants rewriting probably needs new capabilities to reflect updated user requirements and to accommodate current business practices. New capability can be incorporated into, and obsolete functions can be removed from the model. A common revision is to upgrade the specification model to accommodate technology advances.

4.3.3 *Forward Engineering*

Forward engineering is the generation of structured code once the application's functions are identified and the revisions are completed. Re-engineering can be done manually or with automated CASE tools. One goal is to try as much as possible to *reuse* rather than *reinvent.*

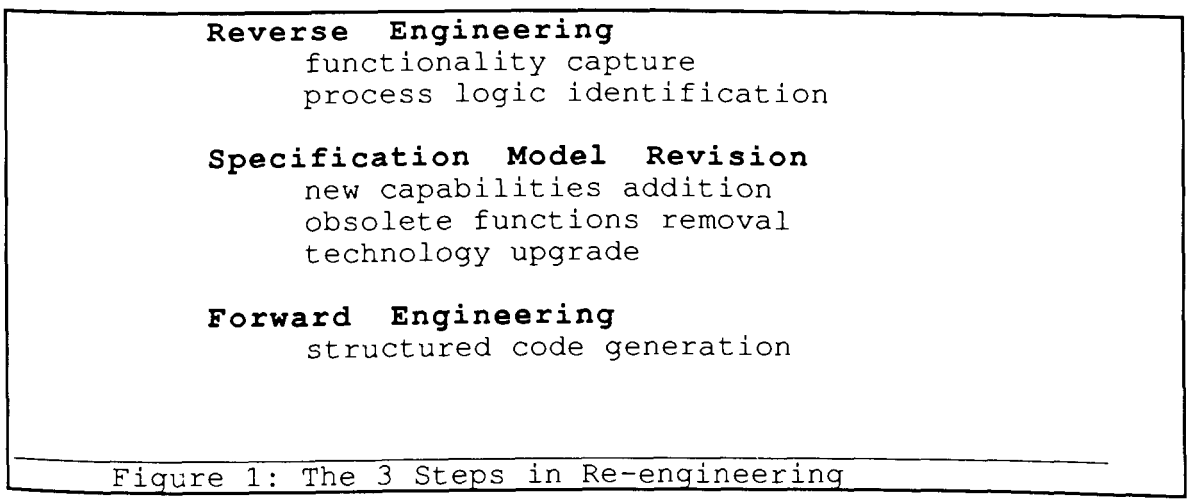


Figure 1: The 3 Steps in Re-engineering

4.3.4 *Re-Engineering Methodology and Tools*

Structured retrofit is an automated method proposed by Lyons and deBalbine in the late 1970s for introducing structured techniques to existing software systems to improve their understandability and maintainability, thus extending their useful life. *Structured engines* are the first restructuring tools, introduced in the early 1980s to transform unstructured FORTRAN and COBOL programs into their respective structured equivalents. "Studies have shown that restructured programs may be a third to a half less expensive to maintain than their unstructured predecessors." [McC92 P.64,69]

There are two types of re-engineering tools: those for *data*, and those for *logic*. Logic re-engineering is at least one or two orders of magnitude more complicated than data re-engineering. A simple example will make the point obvious. For instance, data re-engineering can be used to modify an existing database and to migrate to a new DBMS. Whereas logic re-engineering involves extracting the process design specifications from source codes and regenerating new codes in structured formats. [McC P.101,111,127] Logic re-engineering tools include: testing tools, metrics tools (e.g. complexity analyzers), program code analyzers (e.g. logic analyzers, data tracer), code restructuring tools, and so forth. [McC91 P.27,36]

The *repository* "is the mechanism for defining, storing, accessing, and managing all the information about an enterprise, its data, and its software systems". Repository is the enabling technology of CASE. It is the cornerstone for building an integrated CASE tools environment that can serve the entire enterprise. It is the control point for the description of the enterprise, its data, and its software system. Repository standards are important because they enable software tool integration and the combining of tools from multiple vendors, as well as information exchange between repositories.[1] [McC92 P.157,158,193,214,267]

Some of the re-engineering tools currently available in the market are: PathVu and DataTec from XA Systems, Inspector and Recoder from KnowledgeWare, VIA/Insight, VIA/Renaissance and SmartEdit from ViaSoft, Pretty Printer from Blackhawk, Scan COBOL from CDSI, the Re-engineering Toolset from Bachman,[2] Structured Retrofit from Peat Marwick, and Recoder from Language Technology. [MAR91] Some examples of repository are: the Manager/MVS (centralized) by IBM, CCD/Repository (distributed) by DEC. [McC92 P.147,180] Many of these tools and technology are still evolving. For example, repository technology may not mature for the next five to ten years. [McC P.212]

---

[1] e.g. *CDIF* (CASE Data Interchange Format) is a language specification to enable the sharing of information between CASE tools. It is an extension of the EDIF standard, developed to share information between CAD/CAM/CAE tools. [McC92 P.194]

[2] Analysis tools : PathVu, Inspector
Restructure tools : Retrofit, Recoder
Code splitting tools : VIA/Renaissance [McC92 P.152]

4.4 Differences Between Re-engineering and Other Systems Projects

From an overall project management perspective, there is little differences between systems re-engineering and other regular systems projects, other than the fact that new technology plays a prominent role, implying considerable capital investments and much higher risk.

Naturally, systems designers involved in re-engineering projects which are parts or phases of a BPR must maintain a holistic view, so as not to lose sight of the big overall picture. However, once each such project has been properly scoped, it should basically behave like any other systems project.

As for the development aspect, re-engineering requires three branches of expertise, rather than the normal two (users and IS staff). With re-engineering, the IS personnel come from two camps: participants for the old system, and those for the new system. Since these represent two ends of the spectrum, interactions and coordinations are far more demanding.

As systems re-engineering is still at a very early acceptance stage, it is especially vulnerable to resistance. It tends to be more susceptible to even minor setbacks that may affect but fail to plague the other projects.

4.5 Mission Critical Projects

Over the years, as applications are built upon applications, with data passing from systems to systems, one can no longer modify one application without affecting many others. "It is estimated that nearly $2.5 trillion has been invested in the creation of more than 100 billion lines of code that support current business application systems worldwide. ... The estimated replacement cost for this code is approaching $2 trillion. ... Many companies view their existing systems as valuable asset that is to be preserved, not thrown away. They want to extend the lives of these systems by improving their structure and documentation and upgrading them to utilize new technologies." [McC92 P.3,7,62]

To harness maximum benefits, re-engineering tends to be targeted towards systems which make up the core of the business operations. [BRO92] Such systems often involve mainstream input being massaged and dispersed amongst multiple systems as output. "Re-engineering such systems is often a viable alternative to replacement because it is cheaper,

easier, and safer. The cost of a manual rewrite is $10 to $25 per line of code; the cost of re-engineering lies in the range of $0.02 to $2.00." [McC92 P.29]

In order to minimize the disruptions to other systems feeding on the targeted one, some re-engineering efforts tend to be limited to the *front-end*, leaving the *back-end* intact, so as to maintain normal interactions with the other systems. This is the so-called *wallpaper* approach, which changes the system look, without significantly changing its structure. For example, X-terminals are installed to replace IBM 3270's, with GUI (Graphical User Interface) presenting all the appearance of high technology. It is sometimes described as "putting lipstick on a pig". Because the mainframe software is unaware of the PC-based front-end, it does not take advantage of the pre-edited data and re-edits the incoming data all over again, thus wasting CPU cycles. [BLO92]

These attempts amount to mere window-dressing, marring the true meaning of BPR. However, with such *mission critical* applications, it is understandably justifiable why a thorough overhaul cannot be carried out without jeopardizing or sacrificing years and tens or even hundreds of million dollars worth of software investments. Many more conservative organizations are just too reluctant to risk possible catastrophic failure for longer term unquantifiable intangible benefits, especially considering the failure rate cited in Section 4.2.

4.6 Scope of Analysis

As re-engineering encompasses such a broad spectrum of disciplines, commanding virtually boundless discussions, this paper shall concentrate on a narrower area, restricting the scope to *Systems Re-engineering* alone.

Note that many of the issues to be discussed apply only to very large corporations. It is not the case that bigger enterprises tend to be the ones at the forefront of technology and therefore readily become key players in re-engineering. It is just that smaller companies usually cannot afford the burden of such high bureaucratic overheads imposed by, as well as easily absorbed by the bigger firms. For instance, in a big company, unproductive workers are often tolerated rather than dismissed; whereas a small firm would probably be bankrupted if it retains too many incompetent staff on its payroll.

Moreover, only large organization considering multi-million dollar projects would impose an elaborate decision process to reach a project approval decision. And it is the

very components of this kind of decision process which constitute most of the barriers in our discussion.

Often, the "option to innovate may not be open to many small firms. The high cost of capital investment to automate processes or to introduce equipment that will raise the quality of the products prevents some firms from adopting this strategy." [BRA86] Thus we shall focus our discussion to *large companies* in particular.

5. Motivation and Hindrance to Re-engineering

As with any sizeable project endeavours, obstacles abound to hinder smooth sailing. Hindrances exist at every phase of a project's life history, in the *Inception Stage*, in the *Decision Stage*, and finally in the *Implementation Stage* and beyond. Re-engineering projects are no different, only more so, due to their high profile visibility, enormous technological investments, relative novelty and ecosocial impacts.

5.1 Inception Level

The main ingredients that motivate the inception of BPR are: management foresight, technology awareness and competitive edge.

5.1.1 Management Foresight

A recent ComputerWorld/Andersen Consulting survey of 203 top corporate officers ranked technical expertise as *the* most desired skill for CIOs (Chief Information Officers). [RAD93]

The implications of BPR can only be conceptualized by a progressive holistic thinker with extensive knowledge and a broad vision to see the big picture. That individual must be in a high position of power to effect such a horrendous change. It takes an enlightened top executive with visions to act as the champion to mobilize and to drive BPR.

5.1.2 Technology Awareness

Without the awareness of the concepts of BPR and the availability of IT, the idea of re-engineering will not be conceived in the first place. This is usually where IS plays the key role. IS staff can contribute a realistic view about current capabilities of new technology and the feasibility of carrying out such ventures.

5.1.3 Competitive Edge

Wanting to be at the forefront of competitions is a powerful motivator behind re-engineering. Frequently, in fierce competitions, the choice is between drastic changes or going out of business. Such drastic changes force a business to resort to exploiting the full potentials of re-engineering.

5.1.4 Summary

Motivation plays the key role in bringing about an idea. Once the notion of re-engineering is conceived, its sheer glamour is justification enough to carry it through to be presented for management or executive decisions, where it awaits its fate. So, motivation is the star performer in the Inception Stage. Barriers tend to materialize in later stages.

5.2 Decision Level

The key components which determine the approval or disapproval of a project are found in information generated from requirement analysis and feasibility study: cost benefit analysis, risk factors and mandatory requirements.

When it comes to re-engineering, the *Requirement Study* is not to identify a need, but to present the necessity to *improve* on some *existing* mechanism already in place, for whatever reasons identified at the Inception Level.

The *Feasibility Study* is carried out to establish that the proposed solution is sound, that the benefits justify the costs, that the IT as well as human resources are available, or at least attainable.

5.2.1 Risk Factors

Weighing the uncertainty against the potential gains is always a tough issue. It is at best an educated guess and at worst a calamity. With re-engineering, the uncertainties on both the costs and benefits are especially high. Although risk factors constitute a big obstacle to re-engineering, unless the uncertainty is alarmingly high, for the sake of dramatic productivity gains, they are often tolerated as an inherent part of new ventures.

5.2.2 Cost Benefit Analysis

One of the major deterrents in any project decision process is prohibitive cost. Unless it is an absolute requirement, such as government regulations, a project is unlikely to be launched unless its benefits, tangible or intangible, more the former than the latter, far outweigh its costs, within an acceptable payback period. Especially with re-engineering projects, where risk and uncertainty are assumed and conceded as inevitable, in order to reap the benefits, the major concern that tips the scale is often *costs*.

5.2.3 *Mandatory Requirements*

When the requirements are absolute, the choice is no longer in the hand of the decision makers. Some examples of such mandatory requirements are government regulations, customer expectations, competitive forces and so forth. Such forces, usually external, are often strong motivators to re-engineering.

5.2.4 *Summary*

The fate of a project is often cast in the Decision Phase. Especially for a large size project, barriers and motivators are usually uncovered during the analysis done to ascertain its benefits and probability of success before a project is approved or rejected.

5.3 Implementation Level

The most important contributors to support any re-engineering project are: information technology, human resources and organizational design. [SCH93]

5.3.1 *Information Technology*

As noted in preceding sections, IT is a major enabler to BPR. Whether the technology is robust enough to support the infrastructure remains to be proven.

5.3.2 *Human Resources*

New technology demands specialized skills, which usually involve extensive training. Waiting for the maturity of human resources to catch up with a newly born technology can severely handicap a re-engineering project.

5.3.3 *Organizational Design*

Organizational design affects employee attitude and behaviour, which play a significant role in productivity and effectiveness, thereby impacting the success of a project. We shall explore many of these issues under the sections on Productivity and Employee Attitudes.

5.3.4 Summary

Information technology and manpower are the two key components to sustain the Implementation Stage. Re-engineering is a transformation which depends heavily on IT tools and these tools require people to empower them. Their importance as participants in this stage implies that any factors affecting them (with organizational design being the most predominant) can greatly impact the development of the Implementation Phase.

5.4 The Chief Culprit

However, one must observe that by the time a project is approved for *implementation*, the IT and human resources aspects must have been deemed adequate. Therefore, any difficulties encountered during development are merely day-to-day wrinkles expected to be ironed out, shying in significance to the hurdles of *decision* processes which make them possible in the first place.

Thus, our claim is that the biggest barriers to systems re-engineering surface at the Decision Stage. Barring extreme risks, a large part of the decision hinges on *cost/benefit ratios* and that is where our attentions will be focused on in the upcoming chapters.

6. Barriers Constituted by Cost Factors

Since cost is one of the most critical factor in determining the fate of a project, we shall devote considerable time to analyzing any factors that may impact project costs. Normally, the cost of a project is calculated by multiplying its estimated labour time by a computed charge out rate. Labour time estimation is continually refined as the project proceeds from requirement phase (in man-years) to analysis phase (in man-months) to design phase (in man-days or man-hours).

6.1 Estimated Labour Time

Direct labour requirements go far beyond the normal time for coding and testing individual modules, plus system integrations. For instance, what gets clocked to chargeable hours may be filler times while waiting for users or resources availabilities, or even management indecisions. Some of these factors may be attributed to poor planning, while others can be totally beyond anyone's control.

"In addition, in a study based on over 7000 observations of a group of production programmers, Stalnaker reported that 35% of the time was lost on 'personal activities,' 'being away or out,' and other 'miscellaneous' non-project related activities. In the remaining 65% of available working time, there were further losses spent on mail, non-project related company business, and so on." [ABD91 P.85] The figures agreed well with a previous study done in 1964 by Bairdain of Bell Labs. [FAI85 P.13] However, for the sake of convenience, and the undesirability of otherwise appearing to be non-productive, most of these hours usually get bundled up into time *chargeable* to a project.

So. instead of buffering an project estimate with say 30% extra contingency, any experienced project leader can appreciate why an estimation would not be realistic unless one doubles or triples the idealistic costs. This is how the unproductive part of direct labour get incorporated into chargeable cost, through the buffering syndrome - one version of "*firm cost*".

> *Westheimer's Rule*:
> "To estimate the time it takes to do a task: estimate the time you think it should take, multiply by two and change the unit of measure to the next highest unit. Thus, we allocate two days for a one hour task."

6.2 Computed Charge-Out Rate

The average charge out rate of IS (Information Services) labour for a large corporation is about $50-70 per hour.[1] The average system professional does not earn nearly as much - it is more like $15-30/hr, or $20-40/hr with benefits factored in. So the rest is all *plain* nonchargeable overheads, such as management, systems support, computer operations, hardware, software and nonproductive time (e.g. training and sickness) etc.

The charge out rate generally includes direct labour cost plus all associated overheads such as management and support staff (indirect labour) as well as hardware costs. In its simplest form, such rate is computed by the annual departmental expenses divided by the total chargeable hours for the year.

With the charge out rate being such an all-encompassing item, it is no wonder that any conceivable factor has an impact on its magnitude. This is how such indirect labour and overheads get embedded into the chargeable cost, through dispersed absorption (thus hidden) - another version of "*firm costs*".

Figure 2 depicts a graphical illustration of the size of true cost relative to total cost. The whole block (volume) represents the total chargeable costs for all the projects of a firm; whereas the smaller shaded block represents the absolute direct development cost of project 1.

Actually, firm cost should have both a longitudinal (chargeable rate) as well as a latitudinal (chargeable hours) aspect. But for the sake of clarity and simplicity, the latitudinal component is omitted in the diagram.

---

[1] The writer's survey of three of the largest firms in Vancouver, B.C.:
I.C.B.C.: $60/hr.
B.C. Tel.: $100,000/yr. ($64/hr. assuming 1560 productive hrs/yr)
Westech: $40/hr for programmers; $60/hr for analysts; $100/hr for managers; plus overheads

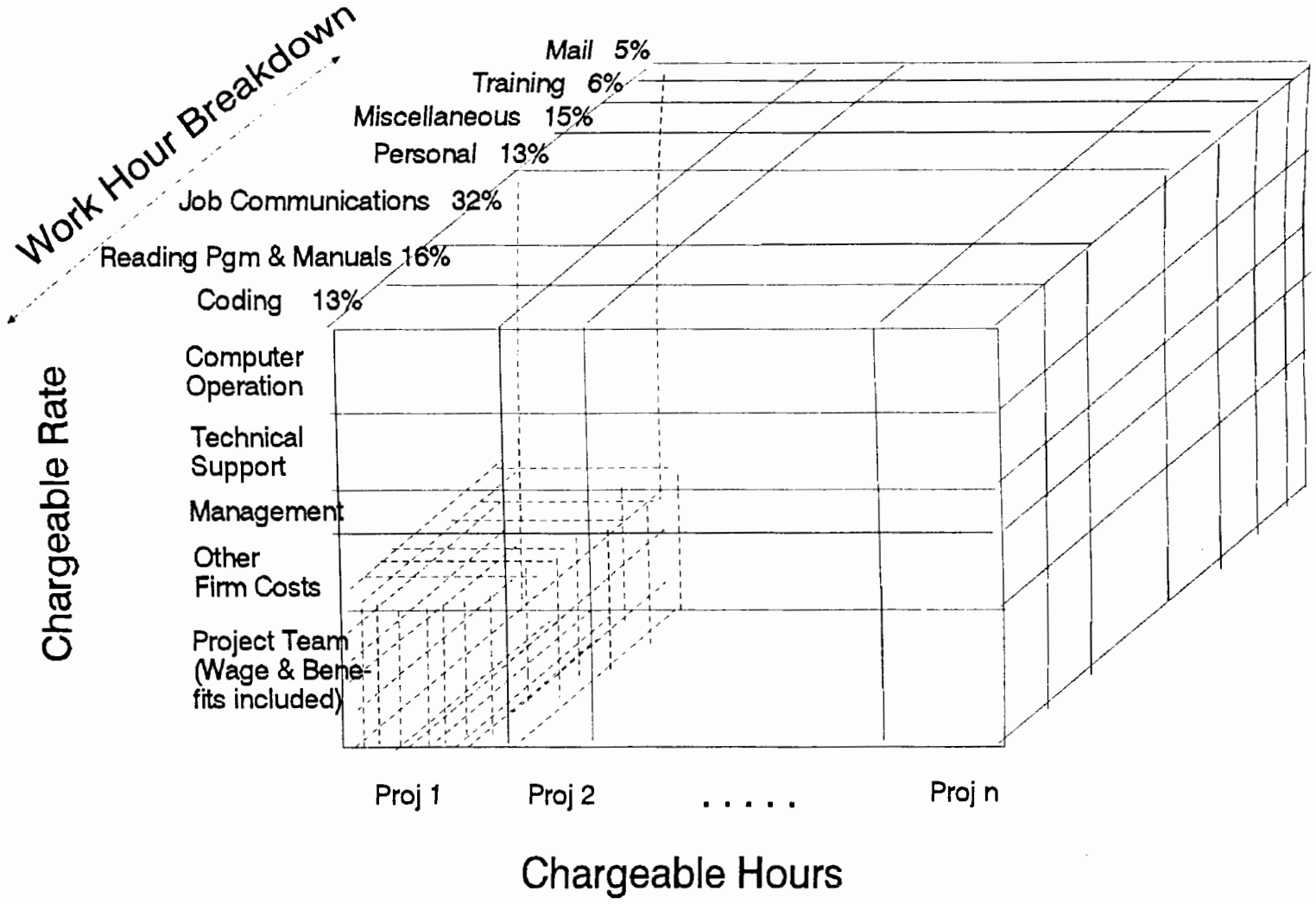


Figure 2: Chargeable Costs of Projects

6.3 Activity-Based Cost Management

Nowadays, hardware costs drop daily while software costs are gradually creeping up. [CUT90] "... software costs are rising, not simply because of the difficulty of developing software systems, but mostly because of the difficulty of maintaining a growing number of critical, existing systems that must be frequently changed to keep up with new business requirements." [McC92 P.5]

Furthermore, beside the steep rise in programmer salary, more and more overheads are imposed on systems development, overshadowing the gain made from productivity improvements introduced by software engineerings and technology tools. One of the most prominent of these "overheads" is probably QA.

Quality Assurance (QA) dictates that every systems project be subjected to close scrutiny by steering committees, technical reviews, team walkthroughs, operations standards and user acceptance. Documentation standards further impose the generations of reports on requirement studies, costs/benefits as well as feasibility and risk analysis, business case studies, systems analysis, technical design, systems and user documentations, and so forth.

Granted, all these requirements should reap paybacks over time, in the form of better systems maintainability, and shorter analysis/development time for future dependent projects, they are nevertheless not *subtracted* from the original development cost. Once a project is installed and delivered, the book is closed and that project carries its price tag forever.

Whereas in the old way of implementation, without the burden of such QA, development cost was at least one-third lower. Although subsequent maintenance costs may negate that savings many times over, they are nevertheless not *added* backed to the original price tag, but rather are charged out to an entirely separate account.

Since subsequent costs are never *added* to the old way, and future savings are never *subtracted* from the new way, software development cost is perceived as ever increasing. Moreover, the figure is considerable with software maintenance typically requiring "40 to 60 percent, and in some cases as much as 90 percent, of the total life-cycle effort devoted to a software product". [FAI85 P.82]

An emerging accounting system known as *Activity-Based Cost Management* is designed to address such problems mentioned above. This accounting practice attaches value to overlooked activities such as delays and reworks, spreading them across the board as a percentage of direct labour. [CAR92] This is a step closer to reflecting the true cost of a flawed project. However, the true culprit is still not explicitly identified. What we really need is a means to retroactively add these maintenance and rework costs back to the original project, which created the problems in the first place, so that the seemingly more expensive quality projects may not be deemed so expensive when compared against them.

6.4 Impacting Factors

Many re-engineering systems projects get watered down or never even survive the proposal stage. The most prominent reason is usually the prohibitive cost associated with its implementation. What management fails to realize is the underlining causes for such exorbitant costs, which are implicated by many factors not yet well explored, stretching far beyond the simple fixed and variable varieties.

By analyzing the hands-on experiences in the workings of a true business world, one begins to realize the complex implications of corporate realities: cost-benefit compromises and risk factors, resource constraints, technological limitations, management and union politics, corporate culture, departmental rivalry, staff turnover, company reorganization, employee attitudes, user acceptance, customer expectations, government regulations, economic climate, and other exogenous forces beyond one's control, all weave a role in this intricate web of a process called *project management.*

6.5 Firm Costs

All these above mentioned issues are what this paper would attribute as factors contributing to "*firm costs*", a proposed term which is somewhat a hybrid between fixed costs and variable costs. Firm cost is analogous to firmware in Computer Science, which can be set up and adjusted (variable) like software, but once installed, behaves like hardware, in a regular consistent (fixed) manner.

Since most of these firm cost factors tend to increase proportionally the cost and/or time of a project, they are usually embedded into the variable side of the calculation. However, upon close scrutiny, one should discover that these problems, once in place,

would persist whether a project is carried out or not. In that sense, they behave much more like fixed costs.

Yet this variable-fixed relationship behaves in more or less the opposite fashion from fixed and variable costs. They are *fixed* in the sense that these issues are regular day-to-day occurrences, an inherent part of an organization (e.g. employee attitude). They are *variable* in the sense that these same issues are either embedded as part of the direct labour cost but hidden under some other pretext (e.g. incompetence); or they are somehow perceived as constituents of indirect labour (e.g. systems or technical support) which should be factored proportionally into the pool of project costs. So depending on one's perspectives, firm cost can be either fixed or variable, or it can be neither. We shall explore this further in ensuing chapters.

Through a good appreciation of these firm cost issues, management would have a better grasp of what plagues most projects, one of the most prominent types being systems re-engineering.

7. Issues Under Study

7.1 Productivity

"An unpublished 1964 study by E. F. Bardain shows programmers realizing only 27% productive time." [BRO74] Some of the low productivity is explainable and even justifiable. (see Section 6.1) Others are far from transparent, as we shall examine below.

7.1.1 Incompetence

In a well-known experiment conducted in 1968 by Harold Sackman, it was observed that the "differences between best and worst performance were factors of 6 to 1 in program size, 8 to 1 in execution time, 9 to 1 in development time, 18 to 1 in coding time, and 28 to 1 in debugging time." [FAI85 P.65] Contrast IBM's operating system OS/360 which was developed by 5000 programmers over a period of five years, with Bell Lab's UNIX which was implemented by Ken Thompson and Dennis Ritchie in a couple of years. [BRO74, BOU83]

Incompetence, or more precisely, a lower level of competence, tends to be an integral, though not necessarily a prevalent, part of a large corporation, not necessarily because it hires a higher percentage of incompetent workers, but more due to the fact that it may be less costly to keep them on payroll than to incur the unpredictable political and legal ramifications after dismissing them. Myrt Webb, a Los Angeles-based performance management consultant with 30 years' experience commented that she seldom run into a manager who enjoyed giving reviews - "That's basically because the review is so negatively oriented. It's a confrontational kind of thing, a punishment, and a negative reinforcer." [MON84] "This displeasure of many supervisors to evaluate subordinates, coupled with an ever-increasing fear of legal action, all but guarantees that only the 'truly poor performer' receives an evaluation below 'average' in many work settings" [WRI93]

Incompetence comes in many flavours. Some people simply do not have the aptitudes to be programmers. Some programmers just cannot catch up with the new tide.

7.1.1.1 Aptitude Deficiency

Invariably, there are workers who land at jobs without adequate skills and/or aptitude to perform the programming tasks (see Section 7.4.3 on hiring practices). Some

of such incompetent programmers are known to their peers and even superiors. Their continual existence is usually due to their seniority with the company. Some are simply not very productive; some are just slow; some may take up co-workers' productive time by requiring interruptive assistance; some can even be detrimental to a project by causing costly mistakes which need to be shouldered by the rest of the team members.

There is a wide spectrum between zero and negative productivity caused by these less competent workers. The obvious errors such as abends and system crashes can be identified and fixed quite readily by the more capable colleagues. Other imminent problems are much harder to pinpoint, and are usually the ones that are contributing to those hidden overheads and expanding the estimated manpower requirements of any project. Many programmers write bad codes that are hard to decipher for later maintenance or future analysis on related or dependent projects. Some bad codes are simply due to bad style. Others are bad because they are padded with unnecessary bulk that tends to side-track or even mislead anyone trying to understand their functions.

7.1.1.2 Behind the Tide

Another flavour of incompetence is what is commonly termed "the Peter Principle" where one is promoted beyond one's level of competence. Older generation programmers can be very proficient with antiquated systems such as JCL, OS files and core dumps. But they are completely at a loss when it comes to PCs, windows and 4GLs. According to the Gartner Group: "By 1997, 85 percent of new applications will require a mixture of skill sets not prevalent in today's typical IS organization." [PRI92, BRO92]

These programmers are usually senior enough to be promoted project leaders. They can be a real asset due to their experiences and familiarity of company applications. However, should they take on projects involving new technology, either they are resistant to change, thus jeopardizing the project, or they rely too much on others to perform the tasks, without expanding adequate supervision (due to their own inadequacy with the technology) to monitor their progress. Should they hire the wrong people for the job, they can create a real disaster.

7.1.2 Adaptability

A significant proportion of the senior analysts in a large corporation's IS department are in their forties, an age which is "too old" to learn new tricks, but too young to retire. Of course, *age* is simply a feeble excuse to mask the true underlining reason of

complacency, lack of motivation, aptitude or competence. Their original tools of trade are rapidly becoming obsolete, and being so entrenched in the old mode of technology, these workers find it much more difficult to adapt to the fast growing new trend. For instance, a switch from an IMS hierarchical database to OO paradigm is more than an 180 degree change; it is more in the magnitude of the third or fourth dimension. [PRI92] These individuals are definitely not suitable candidates for re-engineering projects, other than their experience with the old systems to re-engineer from.

"Redistribution of responsibilities in this way (due to technological changes) might cause resistance from traditional CBIS[1] personnel who develop the feeling of job instability, which will eventually deplete employee morale and loyalty toward the organization. All of these will hinder future CBIS development. ... Besides, a lot of programmers are simply not interested in changing the way they develop software. They are trapped by the common human shortcoming of learning one and only one way to perform a task." These problems "eventually developed into serious stress which might turn into an uncontrollable jeopardy." [KWO93]

7.1.3 Lack of Motivation

Lack of motivation may be due to laziness, lassitude caused by personal or family problems diverting one's energy, or negative reaction to remuneration system perceived as unfair or inadequate. Whatever the reason, employees lacking motivation contribute little to productivity improvement. [SAS90, WEL90, JIN93]

Unmotivated workers are seldom interested in training to upgrade their skills. When they do go on courses or conferences, they view them as mini-vacations away from real work rather than means to improve their skills. "If people do not apply the information presented to them in the first week after they attend a class, there is only a 20 percent chance they will ever use the techniques or methods taught. ... When management helps the employees immediately apply their new knowledge and sets new performance standards to support the change, things really happen." [HAR90]

Large corporations can afford to continually purchase better and more elaborate productivity tools, which unless fully utilized, may or may not justify their cost/benefit

[1] I think CBIS stands for Corporate Business Information Services

ratios. Unfortunately, such utilization and paybacks are neither monitored nor measured. The tools are simply factored into the overall charge-out costs.

Workers with complacent attitudes tend to remain with old tools or techniques they feel at ease with, rather than attempting with new products unless they are mandatory. Thus, some or most of these efficiency tools end up costing the company dearly to acquire and maintain, but are actually sitting idle most of the time.

"A mechanism for motivation, which is attracting interest in the software engineering field, is 'goal setting'. An experiment by Weinberg and Schulman ... found that each team finished first (or, in one case, second) with respect to their objective. They also found that none of the teams performed consistently well on the *other* objectives." [ABD91 P.50]

"In their drive to stay competitive, companies increasingly reward and recognize employees as part of their total quality program. ... Every employee must understand that ... the rewards they receive are just. Results of a survey ... indicate that recognition for a job well done is the top motivator of employee performance." [STU92]

7.1.4 Productivity Tools

Since it is not always possible to hire exceptional individuals, organizations rely on productivity tools and "software engineering to provide notations, tools, and techniques that will enable programmers of good but not outstanding ability to perform their work activities in a competent, professional manner. By investing in better hardware and software tools, organizations can shift software engineering from a labor-intensive to a capital-intensive industry." [FAI85 P.15] Nevertheless, individual ability and motivation are still the primary factor in quality and productivity.

A studies at IBM reported that adding computer resources can decrease system response time, which in turn improve program developer productivity. Programmer productivity increased sixty-two percent with subsecond system response time. [THA84, LAM84]

Although productivity tools, such as CASE and 4GL, tend to shorten development time, they nevertheless come complete with their own drawbacks, which sometimes more than offset the gain. For instance, implementors, being forced to think within a limited range of options dictated by the tools, often have to twist and bend *special* problems in

order to fit the *generalized* solutions offered by the tools. Not only does this obscure the coding, it makes future modifications and enhancement extremely difficult. Moreover, systems bugs generated by these tools are nearly impossible to trace, resulting in frustrating delay or even more convoluted coding to get around the problems. "Also, CASE is proving to be more difficult to implement in practice than originally expected. it will be the turn of the century before CASE will become a mature technology and fully implemented in most organizations." [McC92 P.7]

The most crucial problem with productivity tools such as CASE tools is that they are focused almost exclusively on automating the creation of *new*, stand-alone systems, rather than towards enhancing or rewriting *existing aging* ones, which are the actual growing trend, to maximize return on decades of software investments. [VAN88] Although reverse engineering tools are available in the market, they are often not adequate enough to meet all user needs. For instance, it is still beyond current state of the art to reverse engineer an entire CSP[1] application into the KnowledgeWare Encyclopedia, a repository by an another vendor. [McC92 P.151]

"With so many alternative approaches in the marketplace regarding software engineering tools and methods, some managers are at a loss to find reasonable criteria to make informed decisions. This is further compounded by the organization's lack of identified goals. The drive to examine each and every tool on the market for the 'best' solution has caused many organizations to go into a form of 'Acquisition Deadlock'. Another new tool entering the marketplace causes a new round of deliberation." [CHI89]

Although they pay lot of attention in the up-front investment of acquiring the tools, most organizations never revisit their use of tools. "By not watching what is happening, management encourages the rise of many pitfalls and misconceptions which impede the effectiveness of tools and methods in practice." "Tools are easy scapegoats and get blamed for a lot of management mistakes and inattention." [CHI89]

"A recent survey showed that only 24% of managers now believe that IT has a better return than other investments." An M.I.T. study concluded that "expenditures on IT capital were less effective improving productivity than any other type of expenditures considered." This is because organizations "focus too much on how technology is used

[1] CSP (Cross System Programming) is an application generator by IBM.

instead of what it is used for. ... Systems development methodologies never challenge why things are done in a company, but instead justify the way they are done." [SCH93]

7.1.5 Systems Maintainability

"It is estimated that 80 billion lines of COBOL source code running on some 77,000 IBM and non-IBM mainframe computers support business, industry, and governments across the world today. The estimated replacement cost for this code is approaching $2 trillion. ... More than $43 billion per year is being spent on software maintenance in the United States. [CAR89, McC92 P.3,62]

In 1983, "Weinberg reported that the top ten most expensive programming errors were all maintenance errors." [WEI83, McC92 P.19] A study conducted by Cooper "showed that the rework required by frequent design changes imposed by the Navy was the major reason for a 500 million dollar overrun." [COO80, ABD91 P.33]

7.1.5.1 Design versus Implementation

"There is a growing recognition that there is a need during development to capture the *rationale* - the *why* that underlies the *what* - behind large and complex computer systems. More precisely, there is a growing appreciation of the cost of *failing* to capture this information." [YAK90]

High level design is usually done by senior analysts who have gone through very deep thought processes derived from their in-depth knowledge and practical experiences. However, the significance of such thought processes are rarely conveyed in any detail supporting documents, which tend to relate the mechanics but not the essence of the design.

Junior or inexperienced programmers usually follow the rules or definitions mechanically without thorough appreciation of the underlining motivation behind such designs. More often than not, the instructions are misinterpreted and the design misconstrued.

As a result, the company ends up with unmaintainable products even though they may be based on perfectly superior designs. Since the technical designers are seldom involved in the later development or product acceptance stages, such problems are rarely detected till years later, when significant inconsistencies necessitate the re-examination of the original designs. This is especially true where the design integrate concepts involving

old and new technologies, as in many re-engineering projects, and the junior programmers only have one-sided exposures.

To avoid such communication breakdown, it is imperative that the logic and reasoning behind the design be well documented; and that the designers be more involved throughout the development stage, as well as be amongst the final reviewers and acceptors of the installed system.

7.1.5.2 Standards and Documentation

Most companies have mandatory standards in place to impose proper documentations. They can enforce the practice by disallowing the system to be installed into production unless the requirements are met. However, it is near impossible to enforce the underlining intentions of these requirements. For instance, programmers may know that the system standards insist that a DL/1 program be no more than 64 KB in size, without appreciating the implication that it is intended to be fitted within one logical page, to minimize swapping[1].

How often does any project, under time-pressed constraints as its deadline approaches, exercises adequate quality assurance and walk-throughs to ascertain that each single module does meet the standards of well structured codes, and proper documentations? Every programmer faithfully include comments in the beginning of each module. Most would even supply relevant explanations throughout the program. Yet, simply having the comments there does not necessarily make them useful. A lot of these comments state the obvious but neglect to explain the *why's* nor to describe in detail the *how's*. Their only contribution is to add to the size of the module - taking up disk space.

Other comments can be even more detrimental by having all the appearances of a detail description, but overlooking to mention the most critical factors. If documentation is non-existent, one would be force to study the codes to identify the problems. But when the program seem to be well-documented, one would probably end up in a wild goose chase, looking everywhere else except the right place, to locate a problem which does not seem to match what has been documented. For instance, subtle details such as "inserting message using the IO-PCB (which includes the lterm name) causes the message to return to the

[1] Such limitation is no longer as relevant as CPU and memory costs drop, and hardware performance improves.

requestor" may not be obvious to programmers inexperienced with IMS/DC[1]. Another example may be putting a copy of the target item at the end of an array as a stopper for a sequential search, to save on a condition testing (comparison) instruction from being repeatedly executed each time the search is iterated through the loop. This omission may seem like an oversight to programmers unfamiliar with the trick.

Standards can only serve as guidelines - they are enforceable merely in appearance, never truly in essence. Documentations are only useful if they are produced out of self-motivation, rather than imposed by mandatory standards. According to M. Zelkowitz of University of Maryland, "Problems arise when standards, as a way of coalescing divergent views of a given field, start to lead - that is, when promulgation of standards takes on a life of its own". [ZEL92] The habit of good documentation should have been bred into programmers starting right from the very first day of training, and continued all through their education and career. It is a very long term investment indeed. "The real monetary value of good documentation begins downstream in the development process during the testing phase and continues through operations and redesign." [ROY70]

7.1.6 Other Factors

Of course, many other factors, such as employee attitudes, politics, staff turnover and so on, also play major roles in productivity, directly or indirectly. We shall examine those issues in subsequent sections.

Poor productivity either extends development time (relating to direct labour), or bumps up charge out rate (relating to indirect labour), or both. By lessening or eliminating these productivity problems, project costs can be cut down substantially.

7.2 Employee Attitudes

7.2.1 Resistance to Change

There are many reasons why some employees are resistant to changes:

(1) They lack motivation to learn the new techniques needed for the change.

[1] IO-PCB (Input/Output Program Communication Block) and lterm (logical terminal) are some of the terminology used in IMS/DC (a DBMS online data communication facility by IBM)

(2) They are used to the old ways, feeling comfortable and at home with them. It is risky to chance on uncertainties.

(3) They have become experts in the old systems and may stand to lose their leading edge by switching to the new system where everybody would be competing on equal footings.

(4) They are threatened by change, worrying that change, with its unpredictable outcomes, will have negative impact on their image and career, or even ultimate *job loss*.

Whatever the reasons, these people's attitude can be detrimental to re-engineering since the very term itself suggests nothing but *changes*! Thus it may come as a complete surprise that any such type of characters may be involved in a systems re-engineering project. Surprising but true, since such a project usually consists of three camps: there is always the user side, and there are both the old system and the new system aspects.

Invariably, for a big project, certain individuals from the user community and/or from the old system would belong to the old camp. They are involved in the project due to their invaluable knowledge and experiences of the existing system and requirements. However, their attitudes usually reflect less than full co-operations, occasionally turning into full scale attacks at the very first sign of minor difficulties, which can be numerous for a sizable project.

Some detriments may take a more subtle form. One can come up with a few less than vigorous excuses to continue doing a job in the old way, even though more efficient tools/methods are readily available, thus not taking advantage of potential productivity gains. It is usually not a major concern if it only involves that particular individual. However, if that individual happens to be a key personnel who has power to enforce the standards, it may affect overall productivity level. For example, in the name of controlling multiple simultaneous updates (which rarely happen), one may insist that all documentations be done on the mainframe, using antiquated script facilities, rather than using WYSIWYG (What You See Is What You Get) document processor on the PCs.

As the name implies, resistance to *change* is the biggest enemy to re-engineering, which is nothing but *change*. Some of these resistances may be alleviated through education, reward systems, management reassurance of future prospects, promotion of corporate culture to innovations. However, the most critical factor is still *self-motivation*.

7.2.2 Morale

Staff morale can have a significant impact on productivity, [HAK93] especially in areas where creativity and innovation hold the critical balance, as in BPR. A demoralized employee can contribute to the project's downfall. Many factors affect morale, we shall only discuss two prominent ones.

7.2.2.1 Management Style and Leadership

There are many types of management style. We would only focus on one in particular. Hard driving managers are usually very dedicated employees who truly believe that their every action is done in the best interest of the company. However, by demanding much but rewarding little, they tend to alienate their subordinates. While these managers see themselves as pushing for the highest output at the lowest possible cost, their staff perceive no merit in going the extra miles for "slave drivers".

Overworking one's staff is usually counter-productive. "Once people start working harder, their 'Overwork Duration Threshold,' which represents the maximum *remaining* duration for which they are willing to continue to work harder, decreases below the nominal value." [ABD91 P.87] DeMarco noted that "people under time pressure don't work better, they just work faster. ... In the struggle to deliver any software at all, the first casualty has been consideration of the quality of the software delivered." [ABD91 P.17]

"And there is another kind who are severe, tough, and hard-hitting. But they sacrifice the loyalty of the people around them. ... These coaches (bosses) rarely have sustained success." [RAP93 P.120] In short, intensive task-oriented management style leads to poor morale. [MIL93] Such hard-driving managers are the worst candidates to lead any re-engineering projects, even worse than the *behind-the-tide* managers.

Productivity is motivated by a proper combination of authority and responsibility. Removing achievement, recognition and responsibility causes poor morale, stifling quality and innovations. [HER90] "A contrast can be found in Japan, where in their automotive industry the ratio of supervision to staff is 1:200; in Detroit it is 1:20!" [LUB92]

"Competent, caring management is internal communications at its finest ... more effective downward communication will stimulate increased ideas from employees." [REN90 P.75] Good leaders provide their subordinates with highly defined goals and relevant incentives. A true incentive is self-liquidating, boosting morale and job satisfaction. [WAG90]

7.2.2.2 Art of Selling Oneself

Without good communication skills, some of the most competent workers with advanced foresight become handicapped in presenting their ideas and visions, which may be excellent re-engineering proposals. Unless an adequate channel of communication is provided, good ideas are limited to only those conceived by top management or dictated by circumstances which necessitate their generation.

On the other side of the coin, there are charismatic individuals who are good at presentation, but lacking in true substances. Their rise in the corporate ladder may be made at others' expense. Their contributions to the company depend much on how well they can recognize the talents of their peers and subordinates, and be able to deploy these talents appropriately to further their own glory.

The frustration of relating to a superior who pretends to understand but actually has little or no appreciation of the problems at hand, is far more exasperating than relating to superiors who simply do not care, demanding that the work be done no matter what. In the latter case, at least the subordinate may solicit sympathy from other colleagues. But how may the subordinate let it be known, without jeopardizing himself, that the superior is ignorant and vain?

More often than not, the situation creates a severe morale problem, where a more competent subordinate bemoans the injustices of working under a less competent superior. Such perceived injustice cause true innovation to be withheld consciously or subconsciously.

True competence can only be proven over time. "Respect, once earned, must be constantly re-earned." [FER92] Managers and executives should be wary of high-flyers with impressive speeches but without genuine respects from amongst their peers and subordinates.

7.2.3 Idealists and Perfectionists

These analysts have very good intentions but their energies are usually misdirected towards goals incongruent to realistic overall project objectives. For instance, besides being exceedingly thorough with analyzing the user requirements, spending ample time observing how the users perform their duties, an idealist would offer more bells and

whistles than the users can ever dream of, thus raising their expectation level far beyond the original project scope.

What these idealists seem to fail in realizing is the fact that every project has resource constraints and time deadlines, which can easily be impacted by any extra features not originally committed. Should that idealist be the original estimator for the project, he may never gain approval due to an infeasible cost/benefit ratio. Should he be the analyst for the detail design after the project has already been committed to a certain cost or deadline, his extra proposals may cause undue delay and cost overrun or even ultimate failure of the project ("because of unconstrained goldplating" [VIT92, ABD91 P.52]).

It is a shame that some of the brightest and most innovative ideas cannot be carried through to fruitation. This loss is especially pronounced in an re-engineering case, since these new ideas constitute the very essence of re-engineering. But that is some of the facts of project reality.

The only solution to such problems is to shift the emphasis from *schedule-* to *quality-driven* development, so that innovations and creative ideas may not be encumbered by political pressure. Since the benefits deriving from quality are usually long term and intangible, this approach may be quite a cultural change for most American corporations, which tend to be short-term profit-focused. [HAR90]

7.2.4 Personality Problem

Invariably, there are a few gurus in a large organization who have difficulties relating to people, thus setting themselves apart from the rest. Such brilliant minds can do wonder if they choose to accomplish a task. Given the right challenge, one can complete a task in days which the average programmer may require months, if he can even conceive it achievable.

The "software wizard syndrome" occurs when management "abdicates its responsibility to a highly trusted software specialist, whose pronouncements are *ex cathedra*. Unfortunately, software wizards, unlike the mythical kind, are both fallible and mortal." [BOE79, ABD91 P.53]

Such talents, if fully exploited, especially in a re-engineering project, have tremendous impact on organizational productivity. But due to personality conflicts, they are best left to work on their own, thus limiting their usefulness to the organization. Other

employees perceive them as high-paying philosophizers who make little contributions, which, if measured in absolute terms, is not entirely untrue.

To harness the potentials of such genius, management must ensure that the problems presented to them are challenging and demanding, as well as allow for *subtle* exceptions to accommodate their individual idiosyncrasies, [HOW89] such that they will not be perceived as favoritism.

7.2.5 Peer Rivalry

"... there is another side that can wreck a team of an organization. That is being distracted by your own importance. it can come from your insecurity in working with others ... It can be a feeling that others are a threat to your own territory. These are negative manifestation of ego, ... Ego in these cases makes people insensitive to how they work with others and ends up interfering with the real goal of any group efforts." [RAP93]

IS Department is probably one of the most competitive area to work in. Some programmers take it rather personally if their peers work faster and harder than them. Instead of trying harder themselves, they become uncooperative or even deliberately plant problems in the system to make their colleagues look bad.

These malicious practices are not limited to co-workers, though far less prominent, they can also be employed by irrational superiors who see them as means to punish "unconforming" but otherwise competent subordinates. It is impossible to tap the collective wisdom of the people closest to the issues when rivalry exist amongst team members.

It is difficult to estimate the damages caused by such malicious workers, since they are usually cunning enough to orchestrate the whole scenario to make it appear authentic. Such practices are especially detrimental for re-engineering projects, which rely heavily on self-motivation and team work. [MAR93] To prevent such damaging practices, a project leader must maintain constant interactions with every team member, be very conscious of any disturbing signs or behaviour, and be ready to pursue all underlining causes for irregularities.

7.2.6 Summary

Employee attitudes bear a direct relationship to productivity. Empathy can be a great motivator. [FER92] Superiors sensitive and quick to react to concerns of subordinates can always turn performance in their favour, thus lowering the hidden costs.

7.3 Risk Factors

Risk factors play a major role in adversely affecting any project decision. In the case of re-engineering, where the technology demand and uncertainty are abnormally high, the stakes are definitely stacked against their favour.

7.3.1 Uncertainty

For many years, managers have opted for IBM over other hardware vendors, even though the Big Blue cost many times more. This is because "one can never go wrong with IBM". Unpredictability is a great deterrent against many worthwhile ventures.

Similarly, management find it difficult to approve projects ladened with risk factors. Technological re-engineering projects are the most prominent examples. With assured high costs, and success uncertain and far in the future, it is extremely hard to justify the launching of such projects. Most project coordinators tend to field test individual components to lower the risk. However, one still cannot ascertain that the integration of all components would result in comparable success ratings.

But just as IBM has fallen from favour (no longer does it enjoy the exclusive status of being the first or even the only choice of systems purchases), management should begin to realize that what once used to be guaranteed security may one day be transformed into imminent failure. Hopefully, such new perceptions would shine a silver lining on the dark cloud of any "risky" BPR projects.

7.3.2 Resource Constraints

7.3.2.1 Human Resources

One of the biggest problem to a systems re-engineering project is in the area of human resource, which usually takes the form of inadequate skill sets. Lack of skill may exist amongst current staff, or even in the general market place, since a skill takes time to

develop, long after the new technology is born. Thus extensive training is imperative to any systems re-engineering venture.

"Personnel costs are skyrocketing relative to hardware costs. Chronic problems in software development and implementation are more frequently traced to personnel shortcomings. Information systems staff sizes have mushroomed with little time for adequate selection and training. It is little wonder that Information Systems (IS) managers find themselves focusing increasing amounts of attention on human resource issues." [ABD91 P.49] "It is not enough to have the required human resources; one must provide one's personnel with goals and objectives." [AUD90 P.56]

Unavailability of a *key personnel* can have a significant adverse impact on a project. Such unavailability may be due to leave of absence, extended sickness or delayed release from previous projects. The former, being forewarned and thus provisioned for, tends to be much less disrupting than the latter two, which frequently come unexpectedly at the worst possible time.

Staff turnover is another one of such problems. Under time constraint of imminent deadlines, few projects can afford the luxury of lengthy transition period for outgoing team members to pass on knowledge to new members, or worse still, if new members require trainings. [ABD91 P.64] "Willoughby estimates that the annual turnover in the DP field ranged between 15 and 20% during the 1960s, declined to about 5% in the early 1970s, and began to rise again by the end of the decade. More recent studies place the annual turnover rate at 25.1%, 30%, and even as high as 34%. As McLaughlin points out, at such rates the equivalent of a work unit turns over every three to four years - no minor matter in a profession where it frequently takes 12 to 18 months before a new employee makes significant contributions." [ABD91 P.50] We shall explore this issue in subsequent section.

7.3.2.2 Technological Limitations

Although technology has made leaps and bounds over the last few decades, it is yet *immature* to sustain a robust infrastructure. Many roadblocks still exists, especially for systems re-engineering, which demands integrations of technologies from diverse areas. For instance, with X-Window systems being notorious for its bandwidth consumption, it is impractical to attempt to provide such access across the WAN (Wide Area Network). If a wide spanning network is involved and response time is critical, one must either find other alternatives or provide patches to compensate for the deficiency.

Since re-engineering relies heavily on information technology, any risk factor related to IT poses a strong threat.

7.3.3 Technology Generation Gaps

Less than three decades ago, technology progressed in a fairly slow and steady pace. There was time for software standards and techniques to evolve and mature in tandem. Many applications have been built based on older generation OSs and DBMSs (Operating Systems and Data Base Management Systems such as IBM's OS and IMS), which are well established with known bugs and limitations. System professionals have since developed techniques and work-arounds to compensate such shortcomings, and system problems can usually be isolated within reasonable time frame.

Then comes the new generation of technological revolution, which sees microchip power doubling and tripling at record rate, prices dropping in unison, and canned software pumping out new versions with added capabilities every couple of years. Since these softwares may communicate and/or integrate with other networks, DBMSs, OSs, file systems and even code generators, any problem encountered is no longer clear cut. It may be caused by some bugs a few systems back, or more likely caused by a combination of obscure errors. Tracing such problems can siphon off much of the desperately needed resources of a already sinking project. [PRI92]

As noted in Section 6.3, with hardware cost forever dropping and software cost forever rising, and systems re-engineering so heavily inclined towards the latest technology, it almost seems logical to hold off development decisions until future hardware advances can *downsize* software costs or eliminate them altogether, such pessimistic reasoning is another impedance to project approval.

7.3.4 Other Exogenous Forces

Most exogenous forces which can drastically impact a project are beyond one's control. Some examples are government regulations, politics, reorganization and economic climate. These are discussed under separate sections.

Since systems re-engineering is a relatively new concept, without the benefits of proven, field-tested formal methodology, most such projects are done on a trial and error basis, which tends to impact the bottom line expenditure, adding yet another negative dimension on top of risks and uncertainty.

7.4 Politics

7.4.1 Union Policies

Some unionized policies and job functions can be very rigid. For example, each field installation job has an estimated time period and the coordinator assigns these jobs sequentially to fill up the installers' daily work schedules. If an installer completes a job later than the allocated time, all subsequent jobs assigned to that installer are delayed, resulting in backlogs. If an installer completes the jobs ahead of schedule, the rest of the day becomes idle because he cannot start on jobs for the next day ahead of schedule.

Not all unionized workers like to idle the hours away but they are not inclined to break the union rules either. "But the union didn't think these things were important. ... It was an example of the players being led by and obligated to follow people (the union) who didn't understand the important issues." [RAP93] These types of policies result in poor services and wasted labour resources, another version of the hidden cost.

As mentioned in the ensuing Downsizing section (7.4.6), union rules which apply to layoff is also not very conducive to productivity. Moreover, the existence of a powerful union makes downsizing especially costly.

7.4.2 Corporate Image and Culture

In a competitive market, it is imperative for companies to maintain a flawless public image. More and more progressive workers are turning to firms with suitable corporate culture conducive to employee personal growth and career development. In order to attract top notch employees and to expand and/or maintain customer base, a company must be meticulous about how its image is being projected.

For instance, a firm would opt for attrition over layoffs, even though the process is much slower and harder to control. Also, many companies provide for expensive training programs, even though the payback is often difficult to measure, especially since there is no guarantee that the well-trained employees may not be snatched by the competitors. And of course, companies embracing BPR definitely project a progressive image.

Policies inconsistent with corporate culture de-stabilize employee morale and expectations. For example, Integral, a payroll consulting organization in California, "believes that its re-engineering mistake was to attempt to change the culture too quickly.

This frightened the staff and caused them to mistrust and resist the new technology and the re-engineering life cycle process." [McC92 P.150]

Practices related to corporate image usually incur high expenditures or even exorbitant capital investment, and their paybacks frequently cannot be quantified or determined in the short term.

7.4.3 Hiring Practices

Many managers equate head count to power and try to amass underlings as much and as often as possible. Such sort of bulk hiring invariably precludes true quality. Some superiors refuse to hire subordinates more competent than themselves, a tell-tale sign of their own worth, and an implicit indication of their staff's. Some workers get transferred to the IS department because their old jobs have been eliminated. Some aspired to be programmers and manage to secure the positions due to seniority, connections or plain persistence. Some job seekers are simply good at impressing the interviewers. These types of hiring practices introduce more than its fair share of poor quality in IS's personnel, a sure way to downgrade productivity and effectiveness.

Therefore, the Personnel (Human Resource) Department should always play an active role in all hiring and internal transfers, ensuring true aptitude and aspirations be matched to the right positions. "Utility analyses conducted in diverse organizational settings suggest that validation information and the use of valid selection devices can result in millions of dollars in annual productivity increases for some organizations.[1] ... Moreover, organizations should benefit from follow-up studies that examine the link between their recruitment practices and post-hire effectiveness. ... Firms that employed more of these practices would have higher levels of annual profit, profit growth, and sales growth than firms that employed fewer of these practices." [TER93 P.29,31]

7.4.4 Staff Turnover

"A 1979 Datamation study showed the rate of DP personnel turnover to be about 28 percent annually. Even in the midst of the recent economic recession, Gray (1982)

---

[1] Some examples of effective staffing practices are: recruiting studies, validation studies, structured interviews, cognitive aptitude ability tests, etc.

Some examples of evaluation tools are: BIBs (Biographical Information Blanks for peer evaluation), WABs (Weighted Application Blanks), etc. [TER93]

estimated DP turnover at 15 percent annually." [BAR83] ComputerWorld reported 15% of IS Departments with "deep" (17%) staff cuts in 1992.[1]

"Estimates for the average assimilation period vary between 2 months and 6 months." [ABD91 P.65] "It frequently takes as long as six months for professionals to become comfortable and respected in new jobs." [CAR85] "Even if the individual possesses the needed technical skills, there is a learning curve involved until the individual becomes familiar with the organization." [BAR83] A team is much more productive when the terminology gap is removed by having all people brought to the same level. [CAR85] Moreover, "each team member must learn the project and overcome the learning curve effect before becoming a contributing team member". [FAI85 P.16]

Massive staff turnover tends to be very disruptive to the working environment, thus affecting morale and productivity, easily leading to cost overruns in projects. [BAR83] Major causes for turnovers are reorganization and downsizing, to be discussed in the next sections.

7.4.5 Company and Departmental Reorganization

During any major reorganization, staff members are reshuffled and projects are shelved. Not only does morale decline, productivity tends to be dragged down with all the commotions and uncertainty. Occasionally, some spirits may be bolstered due to certain expectations of possible promotions or better working environment. However, such cases are usually in the minority.

Many projects are stalled during corporate transition periods, such as major company reorganizations, shift of focus with the change of CEO's (Chief Executive Officers) or VP's (Vice Presidents). Enough investment and resources have been committed such that the project cannot be backed out with a moment's notice or the drop of a pen. Yet due to the size of its budgetary requirements, new executives in transition cannot make a speedy decision for it to go ahead. So the project sits in limbo while still clocking up chargeable hours, thus pushing the payback period further into the future, making an executive decision even more unpalatable.

---

[1] ComputerWorld, Dec.28/92 / Jan.4/93, Vol.27, No.1, P.1

Consider again the case of MBL, (see Section 3.6.2) with its introduction of the case manager position which resulted in a major company reorganization. The success of Mutual Benefit Life's BPR efforts was achieved at the expense of the company's paternalistic culture and employee morale. Not only did the displaced workers felt betrayed, even the ones chosen for the case manager positions perceived the new job structure as eliminating advancement opportunities into management. [TER93]

In particular, as BPR demands dramatic reorganization of the corporate infrastructure, its implementation often result in massive elimination and restructuring of job functions. The ramification are so overwhelming that a company cannot simply downsize without serious considerations to all the immediate costs and future consequences.

7.4.6 Downsizing

Survey of managers attending the AMA in 1989 showed that more than half of the companies experienced downsizing within the past four years. However, in order to maintain a good public image and to placate morale, a big corporation can only downsize via voluntary separation and/or retirement, with generous severance packages. For example, IBM announced plan to cut 14,000 jobs in 1991 and implemented the downsizing through attrition and early retirements. [BRO92] More often than not, the deadwood would cling on to their positions, whereas the well-skilled workers with special experiences would opt for the severance[1] and then turn around to work for the competition - a case of double jeopardies - brain drain plus counter competitiveness.

On the other side of the coin, downsizing through union layoffs result in similar problems. Since job security is measured by seniority, those who are first to go are the lower paid and more energetic juniors. Morale subsequent to the layoff bears a direct relationship to the fairness of the layoff and how it is handled. [BRO92] Thus the layoff, if carried out without any compensating or counteracting productivity measures, tends to result in a sharp drop in productivity/cost ratio.

---

[1] Good performers are thought to be more likely to leave because of perceived enhanced attractiveness in the job market. Poor performers, on the other hand, face impending organizationally initiated severance.

Because perceived ease of movement is likely to be greater for higher performers, they are thought to be more likely than are lower performers, to change jobs in response to job dissatisfaction. [BIR93]

Attrition, whereby positions vacated by employees who resign or retire are not refilled, is probably the mildest form of downsizing. From a productivity impact prospective, it amounts to no more than a permanent *leave of absence* (except possibly for the fact that resignation notices are usually much shorter). However, the process tends to be slow and erratic.

Davenport and Short cautioned that "excessive attention to cost reduction results in tradeoffs that are usually unacceptable to ... stakeholders. While optimizing on other objectives seems to bring costs into line, optimizing on cost rarely brings about other objectives." [McK92] For instance, cutting clerks may end up causing high paid employees taking on clerical tasks. [CAR92]

Unless carefully targeted for the right segment of personnel, downsizing can be a costly but ineffective means to boost competitiveness. [HAK93] It is hard to image how having *less* workers with even *lesser* skills can achieve higher productivity. It is even harder to fathom the logic behind lowering the ratio of the skilled/productive over the unskilled/unproductive to achieve it. "Layoffs are a means to an end; corporate strategy helps to define that end". [BRO92]

7.4.7 Unrealistic Deadlines and Poor Planning

With lengthy projects, by the time the system is delivered, the technology may become obsolete or the original requirements may have undergone drastic changes. The urgency of users' needs imposes tremendous pressure on systems developers to complete the project as soon as possible, resulting in unrealistic deadlines. Increasing the head count does not necessarily shorten the delivery date. "There is quantitative evidence to suggest that development time cannot be compressed below about 75 percent of nominal development time regardless of the personnel and resources expended." [FAI85 P.19] "Thibodeau and Dodson suggest that schedule pressures often result in the 'overlapping of activities that would have been accomplished better sequentially,' and overlapping can significantly increase the chance of errors." [ABD91 P.101] As Brooks' Law states: "Adding manpower to a late software project makes it later". [BRO75, ABD91 P.51,212]

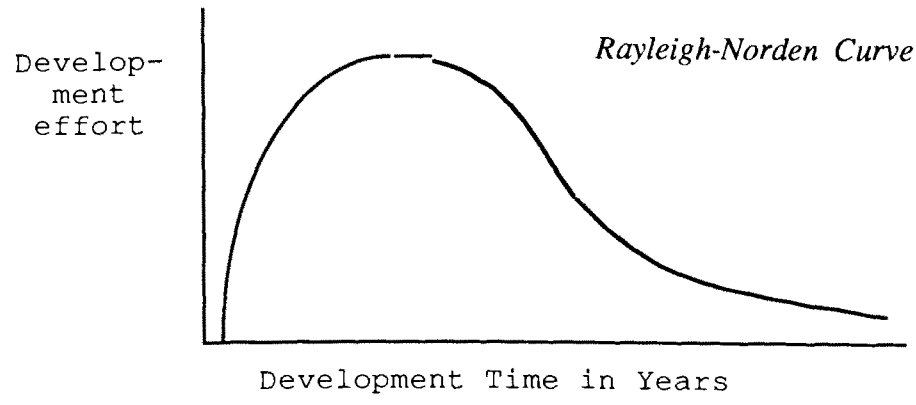
To circumvent this type of unrealistic deadlines imposed by the users, IS (Information Services) should decompose the system into logical parcels, estimating development time and cost for each. End users can then pick and choose the options cafeteria style, to fit into the time frame and goals that they find acceptable.

"In an overly ambitious project, managers who do not understand the details of what they are managing are easily blustered and misled by subordinates. Conversely, low-level staff may be unable to appreciate the significance of details and fail to report serious problems." [ABD91 P.53]

The importance of planning cannot be over-emphasized. For instance, with each IMS region's annual licencing cost running into six-digit dollar figure, such resources are extremely limited. Development teams must book IMS test regions months in advance for production-size testing before final systems promotion. One misses the deadline and the project risks extreme peril.

Appropriate resources must be *strategically* planted in the *right place* at the *right time* to be effective. However, in order to meet unrealistic deadlines promised before the system has even been properly scoped, project leaders amass team members long before the project plan is in place, to counteract the difficulties of securing human resources in short notice. Thus, the project keeps on clocking up chargeable hours while no true deliverables are being produced, a sure road to cost overrun.

The total staff pattern for a project usually resembles a Rayleigh Curve, which climbs steeply at the start; peaks at the heart of development stage; then gradually tapers down; and eventually trails off towards the end. (see Figure 3) [PUT79, JAL91 P.98, PRE92 P.87] By having a resource pool, whereby each project can draw its appropriate team members only when needed, many of the peaks and valleys of staffing can be leveled amongst the projects.



$$K = \frac{L^3}{C^3 t^4}$$

where K= development effort
L = lines of code
C = state-of-technology constant
t = development time in years

Figure 3: Putnam's Estimation Model - The Rayleigh Curve

7.4.8 Departmental Rivalry

Every year, departments compete for their share of the budgetary pie. If a department underspends one year, there is a good likelihood that it will get a smaller allocation of budget funding for the following year. Therefore, it is a common practice for most departments to go on spending sprees at year end in order to cost out any unused budgets for the year, so it is easier to justify for a higher budget request for the next year.

When the supplier could not deliver the goods in time, one organization actually asked the supplier to ship empty boxes in order for the bureaucratic red tapes to register that goods had been received before the year end deadline.

One analyst relates how a department requested for a $50,000 distribution system when all they really needed was a few $1000 printers. The rationale behind the request was that the cost of such a project amounted to internal billing without any real money being exchanged, whereas a printer would had cost the department hard cold cash. Since

the department had enough clout and was allotted the system dollars to request projects as it pleased, the IS department had to oblige. This is just one example of how systems resources are being misused, thereby depriving some more justifiable projects from being launched.

So, instead of being rewarded for frugal practices, a manager feels forced to overspend, sometimes unnecessarily, in order to maintain a steady flow of funding for future contingencies. These wasteful practices are hidden expenses which add unnecessary cost to business operations. This type of issues should provide a strong case to argue for *Zero-based* budgeting where the budget is estimated from the ground up, instead of being computed based on previous years' expenditures.

7.4.9 Fear of Speaking Out

One engineer E relates how his own findings resulted in recommendations against certain proposals which another departmental manager M wanted to push through. E was approached by M to "readjust" his findings and E refused. Subsequently M was promoted to an even more prominent position and E began to realize how many career opportunities he forfeited because he had stuck to his principles.

It is fear of future repercussion such as this which prevents many otherwise upright employees to speak out and causes them to withhold controversial information which may provide the critical leading edge for the organization.

Large corporations usually provide counselling services and opinion boxes to employees. However, few employees take advantage of them due to low publicity, fear of confidentiality violation or lack of trust in the system. To encourage employees to speak out, companies must promote the accessibility and trust of such avenues.

7.4.10 User Resistances

The decision to re-engineer for productivity or efficiency gains is usually made from the top by high ranking executives, without consultation with line staff, the *end users* of these re-engineered systems, resulting in user resentment and resistance, be it overt or covert. Such resistance is especially pronounced in re-engineering projects which tend to involve massive and drastic technological changes.

The VP of Avon Products retained Dr. M. Hammer and four CSC Index consultants to re-engineer his own area in logistics. They spent nine months to design an electronic Avon lady, which was rejected by senior management who were critical of the radical change when they considered the existing process successful. [TER93] It is wasteful to incur the heavy cost of hiring expensive consultants to propose BPR unless the solutions can be carried through and their resulting impacts addressed.

Some causes of user resistance may be perfectly justified, such as ignorance of operation, awkwardness of interface, narrowness of application, etc. "Users are dissatisfied with software systems, not because of system bugs and defects, but because of poor documentation, unfriendly interfaces, and fragile software that breaks when it is changed. Furthermore, software projects fail not because of technical problems but because of a lack of management direction and control." [McC92 P.5]

"In extreme cases, users even experience increased powerlessness when forced to use a rigid, constraining, inscrutable and unreliable computer system - and hence from the users perspective, the very antithesis of the ideal tool." [CLE90] Involving the users all through the project fosters a sense of ownership and ensures a smooth cultural shift as a result of an re-engineering project. That was the experience of Columbia University's five year project. [BAL92]

There are many reasons for user resistance (as cited in Section 7.2.1). Unless the decision makers take these into consideration, these negative attitudes and behaviour are bound to have adverse effects on the ultimate success, even if the re-engineering venture has already overcome all other hurdles throughout the development stage.

7.4.11 Summary

Politics, big or small, tend to affect productivity and organizational effectiveness, contributing substantially to *hidden* costs/benefits. Unless such politically induced problems are stemmed, many individuals as well as the organization as a whole will suffer.

7.5 External Influences

7.5.1 Customer Expectations

To survive modern day competitions, companies must be very responsive to their customers. Not only must their prices be competitive, they should also be constantly aware

of the latest market trends while striving to offer excellent service, which translates to being sensitive to customers' needs, peevishness and expectations.

As Bert Staniar, Chairman of Westinghouse Broadcasting Company, put it:

> "Today's customer is smarter, tougher, and less forgiving than ever before. Today, the customer comes prewired to be cynical, disloyal, and just plain ornery. He has been taught to demand quality, service, and greatness. He hears the words over and over again everywhere, and he's come to see it as his birthright." [HAR90]

For example, the main goal of Mutual Benefit Life's BPR was to improve customer services. (see Section 3.6.2) Also take the case of DOD (Department of Defence, U.S.A.): Proposal reports responding to DOD's RFP (Request For Proposal) may amount to over 7000 pages of text, graphics and pictures, and they have to be submitted within 60 days. In order to speed up proposal processing, one defense contractor had to resort to sophisticated image processing to cut and paste pictorial and graphics documents, with connected database systems to store standard proposal boiler-plate text and original request materials.

One of the biggest criticism on American companies is that they are short-term profit-focused, while Japanese companies work to improve their reputations. "It is much more difficult to rebuild a good reputation than to establish one." [HAR90] And according to John Tschohl, president of the Service Quality Institute in Minneapolis, "extraordinary customer service" will be the key to whether some companies survive or die in the 1990s. [BOZ92b] Therefore, companies should be very sensitive to customer expectations. It is hoped that BPR would prepare businesses to achieve that goal.

7.5.2 Competitions

Competition is the hallmark of a capitalistic society. Even those companies which used to enjoy exclusive monopoly, such as the utilities, are now forced to face competitions. Thus, in an environment of constant changes, as what we are experiencing nowadays, business strategies must be flexible and versatile enough to accommodate all possibilities and eventualities. They should be ready to react at a moment's notice, or face failure.

This is why Ford overhauled their Payables System out of a mandate to sharpen its competitive edge. (see Section 3.6.3) It is for the purpose of competitiveness that Japan

has brought together some of her best minds to develop a Fifth-Generation Computer Technology to leapfrog software business in the U.S. [HAR90]

"Good companies are on their way to bankruptcy, better companies are losing market share, and only the very best are going to grow in the future." [HAR90] And it is generally believed that BPR would make the difference. [SMI91]

7.5.3 Government Regulations

The decision to approve the launching of a project often depend heavily on factors such as cost/benefit ratio, risk analysis, resource availability, payback period, and so on. However, when the requirements are absolute and immediate, such as government regulations, none of the other fore-mentioned factors come into play. The application *must* be implemented. Since manual methods are usually too slow to respond to these new demands, businesses are forced to automate to comply with regulations. One of the most prominent example is the new GST (Goods and Services Tax) being introduced in 1991.

Regardless of costs, many firms hired outside consultants to overhaul their billing systems to incorporate GST in order to meet the January 1st deadline. In one particular company, the consultants uncovered sections of codes that have been in the system for over twenty years, whose functions nobody understood. Some comments seemed to say the exact opposite of what the codes seemed to indicate. The GST implementation is now history. The consultants have come and gone. But those codes still remained in the system because everybody is afraid to touch them. Is this not a likely candidate for re-engineering?

Since such mandatory requirements usually impose a deadline that demands quick fixes rather than permits a thorough idealistic design, *true* re-engineering seldom come into play. Their biggest effect is to divert budget and resources normally allocated to other projects.

7.5.4 Culture and the Environment

In Italy, programmers were told to avoid the term corresponding to "request" because "Only the government requests. Other people 'invite'". Similarly, in Japan, "a direct expression of a request is never (in polite discourse) answered with 'No'" due to their reluctance to offend the listener. [FLO88]

The discrepancy between the presumed culture and the actual culture has resulted in a costly failure of the Employee Evaluation System (EES) developed by Computerized Appraisal Service Group (CASG) for Chemical Company (CC)[1]. Judging from its profitability, CASG presumed that innovation prevailed and risk taking encouraged at CC, and implemented EES accordingly. However, in reality, since the EES information would be available to all levels of managers, instead of the current system whereby the plant managers held exclusive control over determining subordinates' future career, the plant managers perceived EES as a threat to the existing distribution of power, which they wanted to maintain. Under the guise of not wanting to endanger a stable labour relationship, the plant managers resisted EES, as a covert means to oppose the president, who endorsed the implementation. [VIN93]

Companies must find their own niche to compete in the global markets. Wide cultural gaps have made it practically impossible for many North American manufacturers to compete with their Asian or Mexican counterparts, which "enjoy" the low cost of cheap labour, long working hours, poor working conditions, minimal worker benefits and low taxes. Consequently many plants move south or to the Pacific Rim.

Thus the competitive niche for North American firms is best found in high tech areas which demand specialized knowledge, technical expertise and creative innovations. Similarly, individual companies which are positioned in the forefront of technology are much more likely to beat the competitions in the long run, at the cost of short term investment outlay. This is one of the strongest driving force behind re-engineering.

7.5.5 Economic and Social Climate

Businesses no longer enjoy the luxury of the economic and social stability of decades past. During the late 1970s, the economy was booming at an alarming rate, with double digit inflation. Companies went on massive hiring sprees and hiked their capital investments. Consumer confidence and debt loads were at an all time high. By the early 1980s, the boom had turned into a bust. Many of these once high flyers were forced into bankruptcy.

We exist in a dynamic era of constant changes. Our world is changing faster than our ancestors can ever imagine. Computer price is inversely proportional to CPU power,

---

[1] Names have been altered to preserve anonymity.

more exponentially than linearly. Last decade, people are into disposables; this decade, we are into recycling.

We can no longer *anticipate* changes, we may only attempt to be *prepared* to react to them as best as we are capable. Businesses cannot simply plan based on short term forecast. They should be prepared for all long term eventualities. With systems re-engineering being high cost and long time development projects, it is little wonder that managements are wary of their approval, and yet are attracted to them for their radicalism.

7.5.6 Summary

External influences, like politics, though seemingly bear little relationship to direct costs/benefits, nevertheless have very strong impact indirectly, ultimately adding to the hidden costs/benefits. Any ventures done in response to these influences should be perceived as capital investments to strengthen the corporation's competitive advantages.

7.6 Conclusion

All the fore-mentioned issues in this chapter are inherent problems of any organization. The list is by no means exhaustive. These problems usually appear in tandem, reinforcing or counteracting one another, making it difficult to pin point the cause and effect. For example, employee attitude, morale, motivation and productivity are all closely related, seldom existing in isolation.

When it comes to project estimation, these inherent problems subtly assume the form of either overhead or poor productivity, and somehow get incorporated into the chargeable cost. Such over-exaggeration of project cost presents one of the biggest impedance to project approval. Especially with re-engineering, the effect is particularly pronounced.

8. Case Study

Note: Numbers enclosed in double parenthesis ((...)) denote sections referenced within this paper.

8.1 Case Scenario

8.1.1 The Company C

The economy was booming and every business was expanding to take advantage of the growing consumer demands. A Canadian manufacturer C had forecasted a 15% increase in revenue, thus also allocated a matching increase in expenditure. ((7.5.5)) Each department was busy hiring in hope that the authorized increase would be shifted in their favour, as if the budget was to be allocated on a first-come first-served basis. Due to the sudden increased demand, qualified workers were in short supply in the market place. In order to beat the other departments to the budgetary pie, many department heads would create new job positions for fictitious requirements and would settle for less than qualified workers to fill such posting. For instance, the Information Services (IS) Department established two new groups, the Technology Support (TS) Group and the Quality Assurance (QA) Group, increasing the total head count by 30% to 130. ((7.4.3))

8.1.1.1 The Technology Support (TS) Group

The TS Group consisted of seven personnel with an annual staffing budget of $350,000. Its main function was to investigate all new products in the market place and to explore the possibility of applying them within the corporation to improve productivity. Two types of workers were especially attracted the TS Group.

The type X were eager to learn and to be at the forefront of the latest technology. Type Y viewed the position as an escape from the competitive evaluation process where one's performance was measured by the outcome of each project one was involved in. The type X programmers usually worked twice as hard, while the type Y tended to put in only half of their efforts, thus the group still maintained average productivity as a whole, and everything appeared normal to the "outside world".

While the type X were busy acquiring costly new tools for the company, such as UNIX, GUI, client/servers, windows, 4GL and CASE tools, the type Y were only doing a

half-hearted job in assisting the end-users to be proficient at those tools. Except for the occasional few users who managed to procure assistance from the type X programmers, who were more interested in learning than assisting, much of these productivity tool were sitting idle because most people neither knew nor cared how to make use of them. ((7.1.3, 7.1.4))

8.1.1.2 The Quality Assurance (QA) Group

The QA Group consisted of five personnel with an annual staffing budget of $300,000. Its chief function was to set and to enforce all programming and project management standards. Even though the standard proposed by the QA Group were based on sound software engineering principles and would improve systems quality in the long run, most senior analysts viewed it as unnecessary overhead impeding productivity.

By introducing the system life cycle concept, whereby each project, beside the original development and installation phases, was now burdened with four extra formal phases: proposal, requirement analysis, systems design and business case, each phase requiring detail documenting reports, easily adding 30% to project cost as well as extending project delivery date.

The QA's Group's insistence on program modularity and reusability was a precursor to the Object Oriented Paradigm, ((3.7)) which also experienced no more than lukewarm acceptance, since most of the more senior programmers found it hard and unrewarding to change their bad habits and poor programming style. ((7.2.1))

Since no project could be installed without the approval of the QA Group, the development teams felt as if they were raging constant political battles with the QA Group to meet development deadlines, and were resentful of their existence. ((7.4.8))

8.1.1.3 The Economic Downturn

Then, two years later, came the economic downturn and Company C ran into financial difficulties and was eventually bought out by an American manufacturer A. A new CEO E1 was sent to oversee the new operations. As soon as E1 arrived at his new office, he flew in six BPR consultants from a well-known U.S. consulting firm.

Five months and $2 million later, E1 held a meeting with all the top executives to discuss the implementation of the new corporate strategy as recommended by the consultants. The outcome of the marathon meeting session was less than encouraging. All

agreed on the difficulties in implementing the BPR consultants' idealized solutions, which would imply a costly technology investment the current IS Department was neither well equipped nor manpowered to handle, while eliminating many plant and clerical (union) jobs.

Downsizing of 25% not only would result in strong opposition from the union, it would project a negative corporate image for a company once renowned for its stability and job security. ((7.4.1, 7.4.2, 7.4.6))

The parent company A's mandate to align company C's profit ratio in line with its American counterparts was unrealistic, considering the much more stringent Canadian safety standard, ((7.5.3)) bilingual labelling requirements and higher taxes, ((7.5.4)) all adding to the cost of manufacturing. On the other hand, customers expected the same low prices as offered by the American goods, whereas subtle quality differences such as safety features tended to be overlooked.

The final decision reached was to cut the plant manufacturing budget by 10% and to cut the administration/operations budget by 30% across the board, leaving the logistics of cost cutting measures to the device of the individual departments.

8.1.1.4 The Corporate Downsizing

A voluntary separation package was offered to any employee who voluntarily resign. A mini-downsizing was also instituted whereby all junior plant workers hired during the last two years were laid off, as per union policy on seniority. Moreover, since the IS Department was always too short-staffed to implement all those cost-saving projects, it was allowed to absorb many of the otherwise displaced workers, most of whom lack the aptitude to perform adequate programming tasks. ((7.4.3, 7.1.1.1))

Many of the Type X workers in the TS Group in the IS Department opted for the separation package as they had no trouble finding employment with a new spawning industry of software development firms. Their combined severance packages had cost the company over $100,000. Thus the TS Group was now left in the hands of the Type Y programmers, who enjoyed even more slack time, since most users had totally given up on trying to solicit support from the TS Group. As a result, the TS Group existed in name only but still cost the company dearly on staffing, software and hardware. ((7.4.6))

8.1.1.5 The IS Manager M

The IS Department, being the most progressive, was the first to react to the budget cut by launching a major reorganization of its own. One of the IS managers, M, jumped at the opportunity to reshuffle his own staff, separating his favorites from his non-favorites. Programmers and analysts who had ever breathed any sign of discontent were now ((7.4.9)) under the command of supervisor S2, a "slave driver". The rest belonged to supervisor S1, M's very own favourite, who was in charge of all of the high profile projects.

Due to the reorganization, project teams were dismantled and regrouped, causing endless disruptions and discontent. Because each shuffled team member required extra time to familiarize himself with the new project and in some cases even took on extra trainings, many projects missed their milestone deadlines, resulting in delayed systems delivery, countless test rescheduling and unanticipated cost overruns. ((7.4.4))

While all these events were taking place, a pioneer project was at stake. Project P was an innovative re-engineering proposal to overhaul certain aspects of the existing inventory, purchasing and costing systems. It was a watered-down version of one of the many recommendations by the BPR consultants.

8.1.2 The Old System

The WIP (Work In Progress) System was a legacy costing system which had evolved over the past twenty years, interfacing with a myriad of systems. For instance, it obtained the cost of material from the Inventory System's database and provided costing data to the Sales and Marketing System to determine sales prices.

Plant workers entered the material used and labour time spent on work-sheets submitted at the end of the day to the Costing Department, where clerks entered the data into the WIP System. Often, the workers' handwritings were illegible or their hours were not added up correctly, resulting in many system rejects. Any data rejected had to be returned to the workers for corrections and then reentered into the system again. The reiterative process was untimely and error-prone.

Another criticism raised by the BPR consultants concerned the purchasing procedures. Company C owned eight manufacturing plants scattered around the province.

Each plant reordered its own stock, thus missing out on the opportunity to take advantage of the volume discounts centralized purchasing would have provided.

8.1.3 The Proposed Re-engineered System

The BPR consultants suggested that a centralized Purchasing Department be established in the head office, where experienced buyers could negotiate the best possible price amongst the competing suppliers. But since the head office was physically removed from the plants, a remote on-line inventory control networked system had to be set up to trigger stock reordering.

A new system WIPX was also proposed to "replace" WIP. It would require terminals to be installed at various plant sites, where the plant workers could enter the chargeable material and labour directly into the system, which would perform interactive validation, thus eliminating many previously required intermediate processing. As an additional selling feature, WIPX also included a simple Expert System to compute cutting patterns for sheet materials to minimize wastage. ((3.6))

But in truth, WIPX was no more than a front-end interface to the core engine of the old WIP System. By leaving the core intact, the capital investments in the many systems feeding to/from WIP would be protected.

The WIPX and the Inventory Systems would interface with each other, involving client/server technology, whereby any stock taken out from the client plant was updated by the inventory database server.

P was a project of a mixed bag. It included some *reverse engineering* ((4.3.1)) to analyze how the old WIP would be properly interfaced. It also involved addition of a brand new Inventory System in the *specification model revision.* ((4.3.2)) So the final *forward engineering* ((4.3.3)) would consist of a combination of old and new. P was heralded as a *Mission Critical Project* since the legacy system (WIP) it was meant to replace was fed from and producing data for many other systems. ((4.5))

Client/Server architecture would be a pioneering venture for company C. Countless vendor products were studied to evaluate the feasibility and cost of the proposed solution. ((7.3.2.2)) Since most of the products were new in the market place, their reliability and suitability could not be established. ((7.3.2.1)) Also extensive training sessions were necessary to bring the current personnel up-to-speed with the new technology. Although

the project claimed many benefits, most of them were nevertheless intangible; ((5.2.2)) and whatever could be quantified were but at best estimates with high uncertainty. Initial estimate for the project was pegged at $2 million plus equipment costs. Project P was proposed mainly on the strength of the champion leadership of the CEO E1. ((5.1))

8.1.4 The People

As events unfolded, E1's executive leadership was continually being questioned, after successive failed attempts to bolster profits via unprecedented costly re-engineering measures, which in the short term, increased rather than decreased expenditures. E1 was eventually replaced by E2, who appeared to have a more conservative management style. P was then at the project approval stage, the steering committee stalled at the decision pending better appreciation of the new CEO's strategic plans. Meanwhile, team members were still charging their idle time to the project. ((4.2, 7.3.3, 7.4.5))

8.1.4.1 The Leader L

Analyst L had been enjoying a steady rise in the corporate ladder. He was conversant with all the latest buzz words in the computing industry and could articulate very ingenious ideas. Both manager M and supervisor S1 were very impressed with L's presentations and he was promoted to project leader to spearhead project P. ((7.2.2.2))

Before the project was yet to receive final approval, L was already busy amassing team members, knowing from experience that human resources would not be available when needed unless they were retained as soon as they were released from one project but not yet moved on to the next. As new members joined the project team, which was yet to be assigned any real development functions, more idle times were being charged to the project. Before any real work was done, project P had already incurred a cost of $275,000.

In order to orchestrate a convincing case for project approval, L had to juggle the costs and benefits long before all the technical details could be thoroughly assessed. ((5.2, 7.4.7)) Since the executives were the ones who controlled the purse strings, L devoted all of his energy appeasing their concerns, thus neglecting to court the end-users of the system, who were not even aware that their job functions would be drastically altered or in some cases even eliminated. ((7.4.10))

Although L was not particularly strong technically, he was extremely good at recognizing talents. Even though he could not recruit externally due to the hiring freeze, he managed to persuade some of the best talents within the department to join project P.

8.1.4.2 The Designer G

G was touted as the guru of technology within the department and was assigned as the chief design architect. However, being a loner, he did not relate well to people and tended to overwhelmed others with his technical jargons. The development team perceived him as arrogant and difficult to work with, thus were not particularly keen at discussing his ideas. ((7.2.4))

Documentation was also not one of G's passions, and he completed it halfheartedly, filling out the mechanics of the design, without dwelling into its essence. For example, he envisioned that one day many heterogeneous systems would be communicating with each other and a standardized protocol should be implemented in anticipation for such a network traffic. He documented that each request from a client or response from a server had to be packaged in a generic data packet preceded by a standardized header. He defined each field in the header, such as data format type, packet length, sender and receiver addresses, etc., without explaining its significance fully. ((7.1.5.2)) By then, he was too eager to move on to another more challenging task, instead of remaining in project P to nurse his brainchild.

8.1.4.3 The Programmers

Junior programmers in the development team, not appreciating its technical subtlety and thorough holistic considerations, viewed G's design more as a puzzle to be pieced together, instead of an art to be admired. ((7.1.5.1)) They kept on adding extra fields to the header as they saw fit, thus shifting all the necessary communication information out of place.

By following the mechanics in the less-than-complete documentation, many holes were left unplugged and the internal documentations within their programs reflect these inadequacy, all bulk but no substance. They documented on all the extra fields they had to add to the data header in order to make their part of the system work, without dwelling into why they were necessary.

Not until integration testing did they realize that there were communication problems amongst the various systems and finally traced the errors to the data headers. But

by then, they had already forgotten why the system would crash without those extraneous header fields. ((7.1.5.1)) It took them another two months to resolve the problem, costing the project an extra $55,000.

8.1.4.4 The User U

User analyst U used to be a plant foreman three years ago until being transferred to a white collar position due to health reason. He was well versed with the plant's basic needs but might be somewhat out of touch with the latest day-to-day operations since his transfer. He felt a bit wary about these new technological gadgetries but was looking forward to a promotion after the successful implementation of this hi-tech project.

8.1.4.5 The Analyst J

Analyst J was a perfectionist, well-known for being hard-working and thorough with her work. She was assigned to liaise with U and to document the user requirements. Being an idealist, J was always envisioning the perfect system for the users, suggesting more bells and whistles than U would ever dream possible. For example, J suggested the users might want to view the cutting pattern derived by the expert system to allow for interactive alteration of the computed solution.

U was well pleased with J's recommendations and signed off the proposal readily. As a final touch, J insisted a thorough field study was required to ascertain all minor details which might impact her "first-draft" proposal. ((7.2.3)) However, as soon as L saw the costs involved with J's proposal, which would undoubtedly caused a $200,000 budget overrun, not only did he disallow the field study, he relegated J to a lesser post of programming, and assigned a new analyst N, with a mandate to trim J's proposal to an acceptable level.

U was so disappointed with J's transfer that he lost much enthusiasm working with N. And when he was gently coerced by N into dropping most of the nicer features from the system, he became even more uncooperative. It took all of L's diplomatic skills to negotiate a less than satisfactory compromise. The expert system would design the cutting pattern without any user intervention, costing $40,000 instead of $60,000. ((7.4.10))

8.1.4.6 The Analyst R

Since J had been involved with analysis work for over a year, she was a bit out of touch with the latest development tools. R, being the only "expert" in that area, perceived

himself as the leader of that particular development group. However, being a keen and fast learner, J was quickly conversant with the tools, and in time was able to manipulate the software to perform functions R claimed impossible to achieve. Although J never publicized this small victory, R was secretly determined to undermine J's integrity. And when J completed her modules far ahead of schedule, R's resolution became an obsession.

Before J even got a chance to do the final integration testing, R informed U that J's system was ready for user testing, while at the same time he renamed all the systems libraries, under the guise of test promotions. The test system blew up as soon as U accessed it, and U lost complete faith in the idol he once saw in J.

In the mean time, R "just happened to pass by" and offered to resolve U's problem since how J was still busy with some other pressing concerns. Even though it took less than ten minutes to change the library names, R did not inform U till three days later, hinting repeatedly the efforts he had expanded to trace those numerous tough bugs, and he immediately became U's new hero. Upon U's recommendation, R was promoted even before the completion of project P, ((7.2.5)) while J was lamenting on the lack of recognition for her achievements.

8.1.4.7 The Old Guard O

O was a senior analyst well-versed in the old system. Since G was still busy with a previous project, L had relied heavily on O's estimates to compile his business case as presented to the executive steering committee. ((7.3.2.1))

Not having a firm grasp of all of the implications of the new technology, O's estimates were badly skewed towards his own perception and experiences of much older models, which bore no resemblance to the new. With the latest productivity tools, a lot of the modules would have taken less than a third of his estimated time. However, O had never anticipated any of the problems associated with integrating the various network and communication layers involved with the new system, which more than "compensated" for his over-estimation with the individual modules. Had the estimate been done properly, project P would cost over $2.2 million, instead of the $2 million as put forth by O.

Although O was enthusiastic with project P, he refused to step beyond his old system territories, fearing to appear awkward with the latest tools. He had an in-depth knowledge of the old system and was a great contributor to the reverse engineering side of

the project. ((4.3.1)) However, any team member mentioning the interrelationship between the old and the new system was bound to receive his cold shoulder. ((7.2.1))

8.1.4.8 The Team

Not all team members in project P were champion performers like G or J. Considering the size of project P and the constraints in human resources, L had to settle for junior programmers or analysts unfamiliar with the new technology. Some junior programmers were workers transferred from other departments and did not have adequate systems experience; and many others were middle-aged programmers who were conversant only with the old systems but who found it hard to grasp the concepts necessary to interface with the new system. Even after a heavy investment in trainings on client/server concepts, their development efforts still displayed more of the centralized than the distributed approach, as witnessed in the data header problem. ((7.1.1.2, 7.1.2))

As development work progressed, team members began to notice more and more peculiarities in the old system, which would be difficult to model in the new system. The antiquated spaghetti-coded WIP system bore little semblance of modularity, making it impossible to perform a direct translation into classes of the object-oriented GUI. ((3.7)) Moreover, when they discovered capabilities in the new system which would render many old system features meaningless, they were still forced to incorporate those features in the new system as patches, in order not to disrupt the core processing of WIP. ((2.3, 3.4, 4.5))

For instance, in the old system, because of its very own batch nature, many of the processing were done sequentially. Whereas with the new system's windowing and GUI capabilities, many processes could be run in parallel. [BLO92] But with much of the codes and logic being almost undecipherable, it was impossible to determine which set of sequencing were optional and which set were mandatory. They experienced only limited success in wringing the tangled system through re-engineering tools. [NAS92] In order to model the exact outcome, the developers were forced to forego the advantages offered by the new technology, and opted instead for sequential processing, incurring rather unnatural and awkward consequences, which had to be patched up. ((2.2, 2.4))

8.1.4.9 The Supervisor S2

Judging from the cost overruns and missed milestones so far, M decided to rescue his favourite, S1, from disgrace by transferring the responsibility of project P to supervisor

S2. Instead of feeling the burden, S2 considered it as a true challenge to demonstrate his dedication to the company for which he had worked all his career life. By monitoring every move and progress of his subordinates, he permeated amongst all the team members undue pressure over and above what was already imposed by the unrealistic deadlines. ((7.4.7))

As more and more sickness and absenteeism occurred, the rest of the team had to take on heavier and heavier work loads to meet the deadlines, which resulted in even more sick leaves. Due to the company-wide cost cutting policy, none of the employees received remuneration for their overtime, and resentment loomed. ((7.2.2.1))

The once glamour of a pioneering hi-tech project soon lost much of its appeal. Morale was at an all time low and productivity was adversely affected. Had it not been for the economic downturn, many would have resigned, rather than to work under such a hard-driving supervisor like S2.

8.1.5 The Result

All of the political savvy of M and L combined would have not been enough to savage project P, had it not been for the horrendous investment expanded so far. At least L was able to convince the executives that the project was more than 70% done and that further investment would not be throwing good money after bad.

Eventually, the system was completed and ready for field trial and that was when the installer discovered, much to their horror, that the work benches in the plant were far too high for the placement of the X-terminals purchased. Space was already at a premium, there was no possible way to rearrange the plant setup to accommodate new desk space. J knew better than to remind L that had he allowed the field study, such problems would have been identified long before the heavy investment in the terminal purchases. ((7.4.9))

When the last system was installed, more complaints were reported by the plant workers, in addition to those relating to entering data from an awkward position due to the placements of the terminals. As it turned out, the expert system was directing wrong cutting patterns to some new fabricated material, which had to be cut along the grain instead of in any random direction. And without interactive control by human to compensate for the oversight, much materials were wasted.

Since the most junior plant workers were hired at the same time those new materials first arrived at the plant a year ago, they were assigned to work on the material and became the only ones familiar with its idiosyncrasies and cutting requirements. But these junior workers were laid off during the last mini-downsizing, and nobody else was competent to comment on the best way to modify the cutting program. ((7.4.6))

So the $50,000 expert system was scrapped from system P. Disgusted with the injustice caused by L to put the blame on U for overlooking this particular cutting issue, rather than to take responsibility himself for disallowing the field study, J finally resigned to work elsewhere, thus depriving company C one of the last few genuine talents left in the IS Department. ((7.4.9))

Due to the lack of thorough understanding of G's design, more system problems were uncovered after WIPX was installed, when it had to integrate with the existing systems in the production environment. A high overhead of maintenance cost was incurred before these problems were eventually resolved.

8.2 Cost Analysis

8.2.1 Overall Project Cost

Originally, with operations, management, employee benefits and so forth factored in, considering a 80% chargeable time, the IS charge-out rate in company C was computed to be $65/hour ($100,000/year). But after the addition of new groups such as TS and QA, plus all the newly acquired productivity and technology tools, the charge-out rate was hiked to $70/hour ($110,000/year). So for a medium-large size project of about 15 man-year, like P, the cost was increased by about 10%, from $1.5 million to $1.65 million. ((6.2))

Moreover, with QA's introduction of System Life Cycle mandate, project P's life span was expanded from 15 man-years to 20 man-years. Thus the final price tag was estimated to be $2.2 million, a total increase of over 45%. ((6.2))

Therefore, L was finding it much harder to come up with enough benefits to justify the implementation of project P. Had management been presented with the original cost of $1.5 million, with the understanding that an inherent productivity improvement cost of $0.7 million (firm cost) would also need to be factored into the project, as a proportional sharing of such "capital" investment, they might have realized that the payback period for project P was actually much shorter. And if the system implemented successfully by

project P achieved its objectives of productivity and efficiency gains, it would have set a favorable precedence for all future similar projects.

8.2.2 Cost Sharing and Ripple Effect

For the sake of argument, say currently four such projects were approved for development, sharing a total of $2.8 million in firm costs, thus $0.7 million for each project. If all these projects were successful in fulfilling their goals, more such projects would have been implemented. So now six projects would be sharing a slightly higher total firm cost of say $3 million, thus $0.5 million for each project, a considerable efficiency gain.

This ripple effect would continue as long as these projects produce favorable outcomes, and more and more systems are developed to add to the benefits, as well as to share in the firm costs. However, project P had not been well managed, but plagued with problems typical of such projects. The actual underlining causes of these problems usually remained semi-hidden or undetected, and were attributed to other more obvious factors such as technical difficulties and learning curves associated with the new technology, which seemed more defendable with a re-engineering project. By tracing through the case study, we can gain some insight into how these undetected problems contributed to the failure of a project.

8.2.3 Cost Overrun Contributing Factors

By amassing team members long before the project was ready for development, $275,000 (10 men x 3/12 yr x $110,000/man-yr) of non-productive time was charged to the project.

Because of inadequate communications between the designer G and the inexperienced implementation team, an extra two months or $55,000 (3 men x 2/12 yr x $110,000/man-yr) was added to the project cost.

Peer rivalry between J and R added another $20,000 of effort to project P. Even though J's superior performance might have reduced project cost, by say $10,000, R's effort in subterranean sabotage, caused the development team an extra $30,000 to worked through the problems.

O's erroneous under-estimation by $0.2 million, seemingly small (10%) was readily dismissed as unanticipated technological implications, instead of being identified as his misjudgment as it was.

By recruiting less-than-qualified programmers into the project, more time was expanded into training and correcting ensuing development problems, at the tune of roughly $200,000 (6 men x 4/12 yr x $110,000).

The transfer of responsibilities from supervisor S1 to S2, and the ensuing poorer productivity, caused by such factors as sickness and low morale, probably added another $100,000 to the project cost.

So, simply by following through a few isolated incidences, we have already tracked $850,000 worth of wastage and miscalculations in project P, a whopping 38.6%. Since most analysts tend to use previous comparable projects as yardstick, any subsequent projects similar to P would be estimated to cost closer to $3 million than to $1.5 million, thus even harder to justify.

8.3 Case Analysis

8.3.1 Company Problems

In response to the economic boom, Company C attempted expansion by engaging in indiscriminate hiring practices, resulting in a surplus of workers that it could no longer afford during subsequent economic downturn. The correct course of action should have been to expand its work force mainly by hiring casual or part time labour. Its IS Department should respond to the increased systems demands mostly through the use of external consultants and investments in productivity tools. By employing temporary manpower during boom times, stable permanent employment can be maintained throughout the economic cycle, even during financial downturns. By maintaining a fairly steady work force, the company will not be severely impacted by human resource constraints. ((7.3.2.1)) Had it not been for the indiscriminate hiring caused chiefly by departmental rivalry, resulting in excessive payroll costs, company C might not have to resort to downsizing, thereby losing some of its more competent and essential employees. ((7.4.3, 7.4.8)) Consequently, project P encountered problems in recruiting less-than-qualified programmers and inadequate domain experts for the expert system.

As Jackofsky proposed in 1984: "job performance is directly related to job satisfaction and perceived ease of changing jobs", [BIR93] and so are the converses. Some of the best talents hired by company C during boom times, such as those working in the TS and QA Groups, were amongst the ones who took advantage of the severance packages to seek employment elsewhere. Beside the financial consideration, their resignations were also escapes from the job dissatisfaction they had been experiencing. The TS staff slowly realized that their acquired productivity tools had not been well utilized, and the QA staff were tired of being perceived as the enemies against instead of the challengers towards productivity. With their departures, project P lacked adequate qualified human resources to take advantage of the new technology to re-engineer the systems. ((5.3.2))

Capital investment strategies should be planned for wisely and prudently. Establishments of the TS and QA Groups are sound long term quality investments, but they should be viewed as such, not to be deployed as a means to justify a higher budgetary expenditure (as the true motive of the IS Department). ((7.4.8)) The importance of these groups to quality and productivity improvement should be emphasized and promoted under the champion leadership of top executives, so that all staff are aware of their significance. Without such a champion leader to endow executive commitment to the overall operations, the TS and QA groups were doomed to rejections by the more conservative faction in the company. ((3.2))

8.3.2 Departmental Problems

The standards set by the QA Group were sound. QA standards such as walkthroughs "ensured the adequacy, technical feasibility, and completeness of the requirements stated in the logical models and the consistency with previously reviewed logical models." [KNO89] However, first, "at the psychological level, there are actually disincentives for working harder at QA, since it only exposes more of one's mistakes. Second, at the organizational level, there are seldom any rewards that promote quality or quality-related activities." [WEI71, COO79, ABD91 P.103] "It is important that the review process should be treated as a constructive engagement, and should not be treated as means of attacking the ability of the author. Such an attitude by the members can force the author to be defensive, which will defeat the purpose of the review." [JAL91 P.111] Moreover, most of the more established IS staff were not particularly motivated ((7.1.3)) to alter their enshrined programming habits, to accommodate what they considered as the "political

whims" of the "power-hungry" QA personnel. Such type of departmental and peer rivalry is detrimental to the organization. ((7.4.8, 7.2.5))

Personnel working in such groups like TS and QA should be especially people-oriented. This was another one of these personality issues which should not be overlooked. ((7.2.4)) By projecting a cooperative and supportive image, the TS and QA staff would be perceived as working for, instead of working against the users of the facilities, thus averting the alienation persisted within and without the IS Department. ((7.2.4))

Productivity tools should be properly evaluated and acquired based on needs, instead of indiscriminately, as done by the TS Group. Not only did their efforts contribute nothing to efficiency gain, the tools added unnecessary overhead to the department. ((7.1.4))

The type Y personnel in the TS Group obviously lacked the motivation to improve productivity, be it their own, or of those they served. ((7.1.3)) And since there is no concrete mechanism in place to measure their performance, their non-productivity remained unquestioned until the resignations of the type X analysts. By then, most of the end users had lost complete faith in the TS Group created to serve them. So, instead of being a quality asset, the TS Group eventually became a costly liability to the department.

8.3.3 Executive Management Problems

BPR's, no matter how superior in design, are doomed to failure, if done without adequate provisions planned for, to placate the affected personnels. The executive E1 should have anticipated that any BPR initiatives would provoke opposition from the union and other affected departments. ((7.4.4, 7.4.6)) E1's rash decisions and ill-planned strategies had led to his own downfall, jeopardizing the company as a whole. Moreover, a major capital investment, such as required by a BPR, is better accomplished during economic up time, when the company can better afford it, rather than done during the down time, when the pressure for short term recovery tends to overshadow such potential long term benefits, at a tremendous immediate cost outlay. By being ready long before circumstances dictate it, a business is better prepared to react to unexpected customer expectations or the competitions. ((7.5.1)) E1's poor timing of BPR strategies actually cost more than benefitted the company.

As executive E1 was replaced by E2, lower management realized that there had been another shift in executive management style and vision. Therefore they became reluctant to approve a sizeable project conceived by the previous administration, thus stalling project P and causing more cost overrun. ((7.4.5))

The American parent company A should have realized that, due to the cultural differences ((7.5.4)) and incompatible government regulations ((7.5.3)) between the two countries, its expectation of its Canadian subsidiary C to drive its profit in par with its U.S. counterparts was unrealistic. C could not compete with its American counterparts on prices alone. ((7.5.2)) So it should stress its emphasis on quality and services to attract customers. Rather than focusing exclusively on indiscriminate cost cutting measures, it should concentrate more on quality and productivity improvements. ((7.5.1)) BPR projects such as P would have been a progressive step forward, it was unfortunate that it was so plagued with mismanagement problems.

Restructuring is a chief characteristic of BPR. However, such restructuring is meant to achieve an important goal, be it monumental quality improvements or prolific productivity gains. ((3.1, 3.2)) Reorganization for the sake of politics, as carried out by manager M, serves no useful purposes but causes endless disruptions and downgrades morale. ((7.4.5)) His manner of "punishing" subordinates who spoke out against him ((7.4.9)), by removing them from high-profile projects, conveyed a covert message that freedom of expression was strongly discouraged, a stifling blow to innovative ideas, which are so essential for re-engineering.

8.3.4 Leadership Problems

L was superb at selling himself as the ideal project leader. ((7.2.2.2)) But as events slowly unfolded, he turned out to be lacking in both technical substance and project leadership. By amassing team members right at the start of the project, his staffing pattern resembled more of a flat line, rather than the more realistic Rayleigh Curve. Thus much resources were wasted. Due to his own technical inadequacy, he relied on the wrong person O to compile his project estimates, resulting in poor project planning and cost overruns. ((7.4.7)) He put politics ahead of the welfare of the project, ignoring J's recommendation for a field study, thus delaying many critical problem identifications until it was too late to remedy the mistakes (terminal placement and expert system). Especially with re-engineering projects, where much attempts are venturing into the unknown, any

steps carried out (such as the field study) to ascertain a better successful outcome should be encouraged rather than ignored. ((7.3.1))

L's leadership frustrated better workers like J. Not only did he lack proper appreciation of J's concerns over such issues as field studies, L also failed to recognize J's above-average performance, partly due to the malicious interferences by R. ((7.2.5)) So he demoted the champion (J) and promoted the undeserving (R), thereby demoralizing his subordinates, a sure way to downgrade productivity. ((7.5.2)) Beside denying responsibility for his own misjudgment, L also passed on the blame to a user ill-equipped to defend himself due to inexperience. As Winston Churchill once said: "The price for greatness is responsibility." His poor attitudes led to the eventual resignation of J and others, thus depriving company C another of the essential work force necessary to implement sound re-engineering projects. Had there been a better leader at the helm, much of these problems could have been alleviated.

By ignoring to court end-user (plant-workers) acceptance right from the inception of the project, relying only on U to represent their needs, L was also geminating the seeds of user resistance. ((7.4.10)) U was three years removed from the day-to-day operations, thus was ignorant of some of the latest implications such as new cutting requirements. The plant workers, perceiving the new system as a threat to their jobs, jumped on any opportunities to complain about it. And thanks to L's oversights, they found many such occasions in terminal placements and cutting patterns. User resistance is especially detrimental to re-engineering ventures.

8.3.5 Team Problems

As a relatively new concept, systems re-engineering especially needs to attain tangible goals, to establish its status in the systems community. User acceptance, being one of the critical factor in measuring project success, is particularly important to systems re-engineering. By having their expectations raised and then taken away, the users would feel deprived no matter how many features are already offered in the proposed system. With systems re-engineering, the opportunity for innovation seems boundless, and J made the mistake of painting a far too idealized scenario for the user, without consideration to the accompanying costs involved. By venturing beyond the scope, she was jeopardizing the project by introducing extra costs and causing dissatisfactions in disillusioned users. ((7.2.3)) Systems personnel should take a responsible attitude in balancing costs against benefits when suggesting systems features to users, so that whatever they have promised

can be delivered within their resource constraints. However, such constraints impose stringent bounds on quality and innovations, as evident in the shortcoming befallen upon the expert system.

G could have been a tremendous asset to project P had it not been for his difficulty in relating to the other team members. ((7.2.4)) Without adequate interactions amongst themselves, his superior design was not properly conveyed, either verbally or via adequate documentations, to the development team, resulting in a poor end product. ((7.1.5)) Due to lack of understanding of the design, the development team produced numerous undetected errors in their programs. Some "errors manifest themselves, and can be exhibited only after system integration". [SHO83, ABD91 P.105] Thus the design eventually became a costly liability, rather than a quality long term asset, as was the intent of re-engineering.

The delayed deployment of G in the project, due to his other previous project commitments, a fact of life in project management, ((7.3.2)) had resulted in having only O to contribute his one-sided opinions to the project estimation, which totally skewed the outcome, causing misleading cost overruns.

The reliance on O's knowledge beyond his realm of experiences is a case of inappropriate deployment of talents. O, as one of the old guards, was valuable to the reverse engineering side of project P. However, by letting him oversee the whole re-engineering process, his traditional thinking of linear and sequential process structures interfered with the parallelism and integration concepts of true BPR. ((3.2)) O was awarded credit far beyond his level of competence. ((7.1.1.2)) His estimates seemed almost on target to the untrained eyes, ($2 million vs $2.2. million) even though such co-incidental match was achieved inadvertently through a balance between over-estimations and under-estimations. Any subsequent project projection using his estimations as yardstick[1] would no doubt result in budget overruns. And worst of all, personnels working on the over-estimated modules would be enjoying credits not rightfully deserved, while those working on the under-estimated modules would be blamed for problems which are not really their faults. However, it is unfair to place the entire responsibility of under-estimation on O. Considering that a re-engineering project such as P was utilizing brand

[1] The predominant estimation method was "estimation based on a similar project" (used in 67% of the projects), followed by "use of a formula" (40%), "expert opinion" (17%), and "crystal ball" (12%). [Note: Some projects combined methods.] [ABD91 P.47]

new, yet immature, technology in untested water, unanticipated problems were bound to be numerous and unpredictable. ((7.3.2.2))

8.3.6 Project Problems

Conventional wisdom such as *specialization* and *division of labour* merits inclusion even in modern re-engineering concepts. ((2.1)) Had L restricted O's estimates to only the reverse engineering portion of the project, and left the model revision and forward engineering portions to more qualified analysts, his project plan would have been much more realistic. ((4.2))

S2's management style tended to demoralize his staff. "To some degree, real sickness may be a consequence of unfairness ..." [MAR90 P.68] So, instead of extracting extra productivity, his hard-driving supervision resulted in negative payback. ((7.2.2.1))

Without a champion leadership to oversee the integrated implementations at the corporate level, project P was at the mercy of the narrow vision of L, who saw the project only in terms of costs and benefits, instead of fulfillment of a long term corporate strategy. ((3.2)) Many of the players involved did not have a clear understanding of BPR and their misconceptions had contributed to the failure of the project. For example, N viewed the expert system merely as an automated cutting mechanism, instead of an important component in a complex holistic inventory processing procedure. By trimming the interactive portion from the system, he rendered the expert system useless at the end. ((3.5))

The Expert System on cutting patterns was supposed to be a star contributor to the few tangible benefits claimed by project P. But because of misjudgment in cost trimming, it ended up being wasted away with a $50,000 price tag, ((3.6)) as well as depriving the project of many fore-claimed tangible benefits.

Project P was a patch-work semi-attempt at re-engineering, far short of the goals set by the BPR consultants. But considering the horrendous investments involved during tough economic time, it was the best compromise possible, or so it seemed. ((4.4)) Although the WIP was a core system, therefore a very important component of the enterprise system, it was nevertheless taken out of context, and being redesigned without total consideration to the overall corporate picture. ((3.1, 4.4)) Certainly, considering the enormity of the enterprise system, it was a realistic approach to decompose the overall operations into manageable chucks and to re-engineer module by module. However,

instead of taking a holistic approach from the top corporate level, viewing business operations by processes, rather than by functions, their so-called restructuring strategy was merely to select each critical existing subsystem and to re-implement it with new technology. Their efforts amounted to no more than automation, not transformation. Therefore the "restructuring" (or more appropriately "conversion") reaped but marginal benefits, barely enough to cover the cost of the IT vested. ((3.1, 3.4))

Since they were only modifying the front-end processing of WIP instead of doing a total system overhaul, the "new" WIPX System ended up inheriting many obsolete features from the old WIP, being implemented with many work-arounds and patches, defeating the true meaning of re-engineering. ((2.3, 4.5))

Had project P been successful, it would have become the forerunner to future systems re-engineering, thereby keeping the wheel turning for a dynamic and continuous process, as BPRs should be. ((3.2)) However, its failure struck a severe blow to any further re-engineering ventures, for the failure was camouflaged as a failure due largely to IT (Information Technology), rather than due to people problems, as it truly was.

8.4 Summary

Information Technology (IT) and management foresight were the key enablers leading to the very first step towards BPR at company C. However, the problems attributable to the failure of project P were mostly people issues. Some of these problems had been enshrined long before P's inception. For example, indiscriminate hiring practices and extravagant expenditures during economic boom time, misfired executive strategy, unreasonable expectations from parent company A, and so on, helped to sow the seeds of corporate-wide misfortune.

As project P progressed, more people-related troubles accumulated: poor management, inadequate communications, a mixture of misdirected talent and the lack of it, peer rivalry, less-than-qualified work force, etc., all contributed to its failure. Yet, knowingly or unknowingly, people tend to relinquish the responsibility to the defenseless, namely IT, and the defendable, namely their lack of experience with it. Since IT is an inherent part of systems re-engineering, affecting its every aspect, it makes the perfect "scapegoat" explaining away every problem, thus sheltering the real issues from ever being rightfully identified as the problems. ((7.1.4))

9. Conclusion

9.1 Summary and Contributions

BPR is emerging as a competitive imperative for the 1990s, as insightful companies resort to drastic measures to survive global competitions. Information Technology (IT) plays a critical role - it is the key enabler in many of BPR's revolutionary concepts. Systems re-engineering, which deploys much of IT, may form a significant supporting component in the implementation of BPRs.

Re-engineering is but a specialized case of systems projects. Particularly pronounced are aspects such as the application of IT, the contrast between old and new, as well as their accompanying causes and effects, such as risk factors, resistance to change, and so on. Many different factors can impact the success or failure of re-engineering. Furthermore, these factors contribute to the ripple effect of future re-engineering ventures, since people tend to consider past results as possible indications for future performance.

This paper serves to shed some light on the true impediments to re-engineering. It looks far beyond the most apparent rationalization about re-engineering's heavy reliance on IT, which is new, costly, and therefore controversial. It analyzes why a project is not meeting its goals; what prevents a project from being approved for implementation; and what forces a project to be scaled down, thus falling short of its goals of monumental productivity gain.

This paper contributes to the understanding of BPR problems and solutions through two different angles. Firstly, it identifies the problem areas with a balanced view point between management and programmers, thus providing a far more illuminated perspective than the one-sided approach. Secondly, it introduces two brand new ideas to focus on the problem issues: firm cost and reverted IT view point.

This paper claims that BPR failure is the responsibility of both management and systems professionals. Much of the problems lies in the over-concern with project costs; the bias underlying mission critical projects; the misconception of BPR; the seeming contradictory yet actual complimentary aspects between systems re-engineering and BPR; and the failure to observe the three mottos to re-engineering.

By introducing the concept of firm cost as an additional ingredient to cost-benefit analysis, this paper provides a fresh insight to many misconceptions of project costing, alleviating the over-concern about costs, which causes projects to be scaled down or rejected.

Due to the high profile visibility of IT associated with BPRs, many of the above mentioned problem issues dwarf in significance and tend to be overlooked. This paper reverses this improperly skewed perspective and counters that IT is actually playing a supporting rather than a key role to these overlooked issues. This new insight is an absolute necessity for successful BPR initiation and management.

Many re-engineering projects get scaled down due to management's over-concern about costs. However, neither acknowledged nor understood are the assimilated ingredients which constitute the total cost agenda. Many seemingly irrelevant factors are imbedded as hidden costs, which make a project much more expensive than it should be. By tracing through various aspects of a system life cycle within an organizational setting (as demonstrated in Chapter 8), this paper identifies some of these hidden costs (as discussed in Chapter 7), which are built into a project as chargeable costs, even though they are not directly related to actual project development. By bringing them into light, to be viewed under a completely new perspective, these interfering factors (firm costs) will no longer be muddled with the rest of the cost issues. This new approach should open a new way for the decision process, such that a project approval may be determined based solely on its ultimate true costs and true benefits, instead of on poorly defined factors obscured by all these interfering firm costs.

Invariably, these cost issues trace their origin to various aspects of human failing, that is, they are really people issues (as cited in Chapter 7). These cost issues are further implicated by re-engineering, which relies heavily on IT. IT is but a tool to be utilized by human to perform a task. With human factors and IT so inter-twined, there are seldom clear cut boundaries to define the cause and effect of either. With the high profile ascribed to IT, which is itself cost-heavy, the other cost issues tend to be overshadowed. When presenting a postmortem project failure analysis, these human failings are "too numerous, too trivial and too trite" for executive perusal, whereas IT-related problems are not. So IT ends up taking full credit for the cost overrun. Since IT makes such a perfect scapegoat, systems personnels have found for themselves a simple excuse to their own failing. The people issues remain overlooked, intentionally or unintentionally.

Therefore, management is caught in the dilemma of either embracing BPR or denouncing the exorbitant cost associated with IT-ladened ventures. The compromise is usually a scaled down version of re-engineering, which entails a whole new set of problems of its own, mostly as a result of the misconception and poor understanding of BPR by IS professionals. Because of their failure to observe the three mottos to re-engineering (see Section 3.2), IT is used to automate, rather than to transform, thus enhancing instead of correcting the errors. With the people issues unresolved, costs continue to escalate, and IT is seen more and more as the villain which raises project costs to an unacceptable level, while one of the biggest culprits, people-oriented problems, remains overlooked and unresolved. And so the vicious cycle continues to feed on itself.

No single aspect of an organization exists in isolation. Many variables overlap. Many more variables exist as propagating causes and effects. If we agree that IT is the chief enabler for BPR, and that people drive the technology, then the ultimate key is the people and how they are motivated within the organization, which depends on BPR to compete in the 1990s. In essence, there exists a cyclic relationship amongst the four protagonists. (see Figure 4) If an organization cannot foster dedication and competence amongst its employees, it cannot possibly expect the goals of its BPRs to be fulfilled satisfactorily, no matter how powerful the IT it has at its disposal, no matter how well-conceived its BPR design.

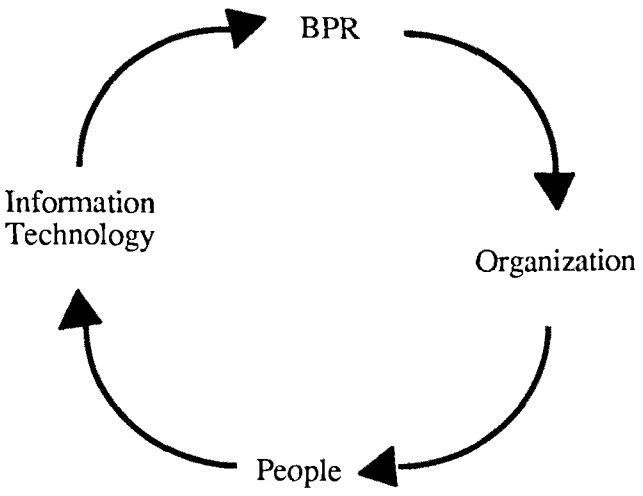


Figure 4: The BPR Cycle - A Proposed Model

The ultimate success of BPR depends on how well the people within an organization perceive the requirements, as well as how well they design and implement the solution. The goal of this this thesis is to provide the readers with some of this insight - the inter-relationships amongst the three function areas of BPR implementation: organization design, human resource policies and information technology. Through a better appreciation of these often ill-understood propagating cause-effect relationships, an organization is better equipped to initiate true BPRs, to improve on the dismal statistics on re-engineering success, to achieve the ultimate goal of monumental productivity gain, an absolute must for global competition.

9.2 Future Work

With the problem areas better understood, we are ready to explore into the area of devising a BPR model of critical success factors, such as organizational culture conducive to innovation and quality, well-defined executive mandate and hands-on sponsorship, crisp channeling of executive vision, clear quantifiable objectives and a firm plan in place, firm commitment of key personnels (such as visionary project leader, innovative designer) to see the project through to completion, motivating development team, in-depth understanding of existing business processes (as well as their pros and cons), encouragement of wondering minds to explore various possibilities for dramatic improvement, thorough knowledge of the latest technology, and so forth.

Although BPR concepts have been gaining popularity since the 1980s, there is yet no product available in the market place to enable companies to disassemble, then reassemble business processes. The Gartner Group, Inc. does not expect serious BPR products to emerge until 1994 or later. [FLY92a] In truth, BPR "is actually 1970s-structured analysis as discovered and repackaged by non-IS personnel". [CAS91] By integrating GUI with entity-relationship, data-flow and related system modeling techniques, one can devise powerful tools to describe, analyze and define BPR models.

Another area of research that is critical to BPR success would be a strategic and tactical model for systems re-engineering. It is a very specialized version of software engineering, since it integrates the old with the new. It involves not merely a firm grasp of the application areas, but also the why's and how's of the old way, so that the new can be applied seamlessly to enhance the good, as well as to eliminate the bad. It demands visionary designers who are daring in their re-structuring tactics; who are not afraid of sacrificing old investments, in order to break away from outdated rules and conventions of

the past, in exchange for superior efficiency; who have the creativity to appropriately marry IT to applications, rather than to force-fit the two. We have touched on many human issues in the thesis, which should form a basis for TQM (Total Quality Management). TQM should play a key role in any labour-intensive endeavour, such as systems re-engineering.

One more area of study that should be of particular interest to BPR is the design of a perfect IT architecture to align with the business infrastructure (or business strategy). IT architecture is defined as "a series of principles, guidelines or rules used by an organization to direct the process of acquiring, building, modifying and interfacing IT resources throughout the enterprise. These resources can include equipment, software, communications protocols, application development methodologies, database systems, modeling tools, IT organizational structures and more". Cornelius Sullivan Jr., president of Information Technology Planning Corp., Chicago, states that the goal of architecture is "to achieve fit or harmony between form and context". [ROS92, FLY92b] For example, if a business wants to be the market leader, it needs an IT architecture that empowers quick decisions. The range of possibilities for such an architecture alignment is enormous, and the task to mould such a discipline should be a real challenge.

Appendix

The following list of definitions are quoted from references cited at the end of each paragraph.

Reusability and Reverse Engineering

A **component** is a closely-knit cluster of classes that act as a unit. [BAT92 P.357] A component can be thought of as a layer, where a software system is a stacking of different layers (i.e. a composition of components). [BAT92 P.358]. Components are not monolithic, but are suites of algorithms that translate data and operations from a component's abstract interface to data and operations of its concrete interface. [BAT92 P.380]

Every component implements an *abstract-to-concrete* mapping, which is a transformation of objects and operations visible at its interface or abstract level to objects and operations at its concrete level. [BAT92 P.358]

Composition is the rules and operations of component parameter instantiation; i.e., the guidelines by which components can be glued together. A *software system* is a type expression (i.e., a composition of components). ... The set of all software systems that present the interface of realm T is called the *domain* of T, denoted Domain(T). [BAT92 P.359]

Algorithm reuse occurs when the same algorithm is used in two different components. [BAT92 P.377]

Conceptual distances between items of each facet are used to evaluate their similarity, which is used in turn to evaluate the similarity between required software specifications and available components. ... Conceptual distances are assigned based on experience, intuition, and common sense. ... Frequencies of "perceived similarity" obtained by running experiments with controlled groups of individuals are used to compute a "**dissimilarity coefficient**". [OST92 P.209]

The **subsumption relation** is intended to capture the idea that certain components can be built by composing several other components. If the functionality of a component A is partially provided by a component B, then B (the *subsumer*) is considered to be a

suitable reuse candidate to construct A (the *subsumed*). ... For example, consider the abstract data types *stacks* and *lists*. The stack operation *append* is subsumed by the list operation *cons*, because *append* can be constructed using *cons* as a subfunction. [OST92 P.213]

In general, to create a library for software reuse it is necessary to perform a **Domain Analysis**, defined by Prieto-Diaz as the process of identifying, collecting, organizing, analyzing, and representing a domain model and software architecture from the study of existing systems, underlying theory, emerging technology, and development histories within the domain of interest. [OST92 P.216]

Every operation of a component's interface is implemented by one, or perhaps several, algorithms. Cataloging these algorithms exposes the potentially complex internal structure of components. Algorithm catalogs explain how variations of components arise in practice, how individual components may be customized to use a particular task, and how 'as is' algorithm reuse can be realized. [BAT92 P.376]

A **model of a domain** is the set of realms and the rules of composition that define the software systems of that domain. It is also a grammar for expressing the systems of a domain as compositions of primitive components. A software system is a sentence and a domain is a language. [BAT92 P.360]

Bibliography

[ABD91] T. Abdel-Hamid & S. Madnick
"Software Project Dynamics - An Integrated Approach"
Prentice Hall, New Jersey, 1991

[ABD93]* T. Abdel-Hamid
"Adapting, Correcting, and Perfecting Software Estimates: A Maintenance Metaphor"
IEEE Computer, March 1993, P.20-29

[ALL93]* D. Allen
"Performance Anxiety"
ComputerWorld, Feb.15/93, Vol.27, No.7, P.89-

[AMB92] J. Ambrosio
"The Almost Paperless Office"
ComputerWorld, Aug.3/92, Vol.26, No.31, P.71-

[ANT93] G. Anthes
"Welcome to the Unknown Zone"
ComputerWorld, Feb.18/93, Vol.27, No.6, P.55-

[ARN86] R. Arnold
"System Maintenance vs System Redesign"
Tutorial on Software Restructuring, IEEE Computer Society Press, Washington, D.C., 1986, P.258-261.

[AUD90] R. Audet
"Human Resources: Motivate or Stagnate"
Manager's Magazine, Vol.1, No.3, Spring 1990, P.54-57

[BAL92] M. Ballou
"Assignment: Re-engineering"
ComputerWorld, Nov.9/92, Vol.26, No.45

[BAR83] K. Bartol & D. Martin
"Managing the Consequences of DP Turnover: A Human Resources Planning Perspective"
Tutorial: Software Engineering Project Management, IEEE Computer Society Press, P.338-345]

[BAS91] V. Basili & J. Musa
"The Future Engineering of Software: A Management Perspective"
IEEE Computer, September 1991, Vol.24, No.9, P.90-96

[BAT92] D. Batory & S. O'Malley
"The Design and Implementation of Hierarchical Software Systems with Reusable Components"
ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 4, October 1992, P. 355-398

[BEN82]* C. Bentley
"Computer Project Management"
Heyden & Son, 1982.

[BET92] M. Betts
"Files with Faces - Banc One Re-engineers"
ComputerWorld, Dec.14/92, Vol.26, No.50, P.93-

[BIG89]* T. Biggerstaff & A. Perlis
"Software Reusability I: Concepts and Models"
ACM Press, 1989

[BIG89a]* T. Biggerstaff & A. Perlis
"Software Reusability II: Applications and Experience"
ACM Press, 1989

[BIG90]* T. Biggerstaff & C. Richter
"Reusability Framework, Assessment, and Directions"
Software Reusability, Vol. 1, T. Biggerstaff & A. Perlis, Eds., ACM Press, New York, 1990
(Also in IEEE Softw. 4, 2, Mar. 1987)

[BIR93] D. Birnbaum & M. Somers
"Fitting Job Performance into Turnover Model: An Examination of the Form of the Job Performance-Turnover Relationship and a Path Model"
Journal of Management, Vol.19, No.1, Spring 1993, P.1-12

[BLO92] E. Bloom
"Getting Down"
ComputerWorld, Aug.10/92, Vol.26, No.32, P.69-

[BOE79] W. Boebert
"Software Quality Through Software Management"
Software Quality Management, Petrocelli Books, Inc., New York, 1979

[BOU83] S. Bourne
"The UNIX System"
Addison-Wesley, 1983.

[BOZ92] J. Bozman
"Wrapping Code Can Save Time"
ComputerWorld, Nov.30/92, Vol.26, No.48, P.73-

[BOZ92a] J. Bozman
"AI is Out; Objects are In"
ComputerWorld, Dec.28/92, Vol.27, No.1, P.27-

[BOZ92b] J. Bozman
"Overhaul Ahead"
ComputerWorld, Dec.28/92, Vol.27, No.1, P.29-

[BOZ92c] J. Bozman
"Oracle Conference Goes Look for Proof of Change"
ComputerWorld, Sep.14/92, Vol.26, No.37

[BRA86] T. Brady and S. Liff
"Job Losses Now, Maybe Some Later"
"The Information Technology Revolution" The MIT Press, 1986, P.381-389.

[BRA91a] S. Bradley
"Business Process Automation: Applying Technology"
Gartner Group, Inc., IS Key Issues, K-800-638, Dec.16/91.

[BRA91b] S. Bradley
"The Next Wave: Business Process Automation"
Gartner Group, Inc., IS Key Issues, K-800-637, Dec.16/91.

[BRI83]* A. Brill
"Building Controls into Structured Systems"
Yourdon Press, 1983.

[BRO74] F. Brooks, Jr.
"The Mythical Man-Month"
Tutorial: Software Engineering Project Management, IEEE Computer Society Press

[BRO92] J. Brockner
"Managing the Effects of Layoffs on Survivors"
California Management Review, Vol.34, No.2, Winter 1992, P.9-28.

[CAF93] R. Cafasso
"Re-engineering Plan Preserves Mainframe Role"
ComputerWorld, Apr.12/93, Vol.27, No.15, P.63-

[CAF93a]* R. Cafasso
"Re-engineering Projects: Up, Up and Away"
ComputerWorld, Mar.29/93, Vol.27, No.13, P.92-

[CAF93b] R. Cafasso
"Re-Thinking Re-Engineering"
ComputerWorld, Mar.15/93, Vol.27, No.11, P.102-

[CAF93c] R. Cafasso
"Step By Step, Companies Move Ahead"
ComputerWorld, Mar.15/93, Vol.27, No.11, P.104-

[CAL91] G. Caldiera & V. Basili
"Identifying and Qualifying Reusable Software"
IEEE Computer, February 1991, P.61-70

[CAR85] M. Carpenter and H. Hallman
"Quality Emphasis at IBM's Software Engineering Institute"
IBM Systems Journal, Vol 24, No. 2, 1985, P.121-133

[CAR89] R. Carlyle
"Fighting Corporate Amnesia"
Datamation, February 1, 1989, P.41-44

[CAR92] W. Carlson & B. McNurlin
"Do You Measure Up"
ComputerWorld, Dec.7/92, Vol.26, No.49

[CAS91] A. Case
"Business Process Re-Engineering"
Gartner Group, Inc., SES Strategic Planning SPA-210-590, Aug.7/91.

[CAS92a] A. Case
"Business Process Re-Engineering Questions and Answers"
Gartner Group, Inc., SES Events, E-210-708, Apr.30/92.

[CAS92b] A. Case
"TI Announces BDF to Re-Engineer Business Processes"
Gartner Group, Inc., ADM Products, P-260-773, Oct.5/92.

[CHI89] E. Chikofsky
"How to Lose Productivity with Productivity Tools"
Experience with the Management of Software Projects 1989, IFAC Workshop Series, 1990. No. 9

[CLE88]* C. Clegg, P Warr, T Green, etc.
"People and Computers - How to evaluate your Company's New Technology"
Ellis Horwood, 1988.

[CLE90] A. Clement
"Cooperative Support for Computer Work: A Social Perspective on the Empowering of End Users"
Proceedings of the Conference on Computer-Supported Cooperative Work, Oct. 7-10, 1990, Los Angeles, CA, Sponsored by ACM SIGCHI & SIGOIS, P.223-236

[COO79] J. Cooper
"Software Quality Management"
Petrocelli Books, Inc., New York, 1979.

[COO80] K. Cooper
"Naval Ship Production: Acclaim Settled and a Framework Built"
Interfaces, Vol.10, No.6, December 1980.

[CUT90] A. Cutaia
"Technology Projection Modeling of Future Computer Systems"
Prentice Hall, 1990.

[DAV90] T. Davenport, J. Sbort
"The New Industrial Engineering: Information Technology and Business Process Redesign"
Sloan Management Review, Summer 1990.

[DAY88] U. Dayal
"Active Database Management Systems"
Proceedings of the Third International Conference on Data and Knowledge Bases, 1988, P.150-169

[DRU88] P. Drucker
"The Coming of the New Organization"
Harvard Business Review, Jan-Feb 1988.

[ENG92]* G. Engels, C. Lewerentz, M. Nagl, W. Schafter & A. Schurr
"Building Integrated Software Development Environments Part I: Tool Specification"
ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 2, April 1992, P.135-167

[FAI85] R. Fairley
"Software Engineering Concepts"
McGraw-Hill Book Company, 1985

[FER92] P. Fernberg
"Leadership and Empathy: Wally Bennett Helps Gast Fly"
Modern Office Technology, December 1992, P.38-.

[FLO88] F. Flores, M. Graves, B. Hartfield & T. Winograd
"Computer Systems and the Design of Organizational Interaction"
ACM Transactions on Office Information Systems, Vol. 6, No. 2, April 1988, P. 153-172

[FLY92] S. Flynn
"A Holistic Approach to the IT Infrastructure"
Gartner Group, Inc., IS Key Issues, K-970-890, Apr.30/92.

[FLY92a] Flynn, Raphaelian, Rosser, Smith, Terdiman, Tunick, Vargo
"Managing IT: BRAVO - Bus. Proc. Re-Engineering"
Gartner Group, Inc., IS -R-980-113.4-113.5, May 15/92.

[FLY92b] Flynn, Raphaelian, Rosser, Smith, Terdiman, Tunick, Vargo
"Managing IT: BRAVO - Architecture"
Gartner Group, Inc., IS -R-980-113.6-113.10, May 15/92.

[HAK90] C. Hakim
"Boost Morale to Gain Productivity"
HR Magazine, February 1993, P.46-49.

[HAL92]* M. Halper
"Canion Warns Re-engineering: Beware of Outsourcing Dangers"
ComputerWorld, Nov.9/92, Vol.26, No.45, P.73-

[HAM90] M. Hammer
"Reengineering Work: Don't Automate, Obliterate"
Harvard Business Review, Jul-Aug 1990.

[HAM92] M. Hammer, J. Champy
"What is Reengineering?"
Bonus issue of InformationWeek, May 5, 1992.

[HAR90] H. Harrington, Ernst & Young
"Why Focus on Business Processes?"
TR 90.001 HJH, August 1990.

[HAR90] P. Harvey
"Communicating with Employees: B. C. Tel Does It Well"
Manager's Magazine, Vol.2, No.1, Summer '90, P.26-32

[HEL88]* M. Helander, editor
"Handbook of Human-Computer Interaction"
North-Holland, 1988.

[HER92] F. Herzberg
"One More Time: How Can You Motivate Employees?"
Manager's Magazine, Vol.2, No.1, Summer '90

[HOF93] T. Hoffman
"Keeping Track of Software"
ComputerWorld, Apr.29/93, Vol.27, No.16, P.65

[HOR92] E. Horwitt
"DHL Revamps with Big Blue"
ComputerWorld, Nov.30/92, Vol.26, No.48, P.1-

[HOW89] B. Howard & D. Lebell
"Tapping Technical Talent"
Personnel, November 1989, P.53-

[HUF92] S. Huff
"Reengineering the Business"
Business Quarterly, Vol. 56, No. 3, Winter 1992

[JAL91] P. Jalote
"An Integrated Approach to Software Engineering"
Springer-Verlag, New York, 1991

[JIN93] P. Jin
"Work Motivation and Productivity in Voluntarily Formed Work Teams: A Study in China"
Organizational Behaviour and Human Decision Processes, Vol.54, No.1, Feb.93, P.133-155

[JOH92] M. Johnson
"Bank Touts Revamp as Customers Draw"
ComputerWorld, Nov.23/92, Vol.26, No.47, P.1

[KIN92]* D. King
"Project Management Made Simple"
Yourdon Press, 1992.

[KNI92]* R. Knight
"Fujisawa: Re-Engineer, Then Streamline"
ComputerWorld, Dec.21/92, Vol.26, No.51, P.52-

[KNO89] V. Knowes
"Successful Quality Assurance Plan for a Large Project"
Experience with the Management of Software Projects 1989, IFAC Workshop Series, 1990. No. 9

[KWO93] L. Kwok & K. Arnett
"Organizational Impact of CASE Technology"
Journal of Systems Management, March 1993, P.24-28

[LAM84] G. Lambert
"A Comparative Study of System Response Time on Program Developer Productivity"
IBM Systems Journal, Vol.23, No.1, 1984. P.36-43

[LAN84] R. Lanergan & C. Grasso
"Software Engineering with Reusable Designs and Code"
IEEE Trans. Software Eng., Vol. SE-10, No.5, Sept 1984, P.498-501.

[LAW90] H. Lawson
"Philosophies for Engineering Computer-Based Systems"
IEEE Computer, December 1990, P.52

[LED92] A. Lederer & J. Prasad
"Putting Estimates on Track"
ComputerWorld, Aug.24/92, Vol.26, No.34, P.85-

[LEV87] L. Levy
"Taming the Tiger - Software Engineering and Software Economics"
Springer-Verlag, 1987

[LUB92] J. Lubans
"Productivity in Libraries? Manager Step Aside!"
Journal of Library Adminstration, Vol.17, No.3, 1992, P.23-42

[LUG89] G. Luger & W. Stubblefield
"Artificial Intelligence and the Design of Expert Systems"
Benjamin/Cummings Publishing Company, Inc., 1989

[LUQ89] Luqi
"Software Evolution Through Rapid Prototyping"
IEEE Computer, May 1989, P.13-

[MAG93] J. Maglitta
"Squeeze Play"
ComputerWorld, Apr.29/93, Vol.27, No.16, P.86-

[MAR91] E. Martin, D. DeHayes, J. Hoffer & W. Perkins
"Managing Information Technology - What Managers Need to Know"
MacMillan Publishing Company, New York, 1991.

[MAR93] N. Margolis
"It's Time for Teamwork"
ComputerWorld, Apr.5/93, Vol.27, No.14, P.78-

[MAR93a] N. Margolis
"Re-engineering Gets Real"
ComputerWorld, Jan.18/93, Vol.26, No.3, P.69-

[McC92] C. McClure
"The Three Rs of Software Automation - Re-engineering, Repository, Reusability"
Prentice Hall, New Jersey, 1992

[McK92] J. McKeen & H. Smith
"Reengineering the Corporation: Where Does I.S. Fit In?"
Working Paper 92-12, May 1992.

[MEH93] M. Mehler
"Publix Enemy No. 1"
ComputerWorld, Mar.1/93, Vol.27, No.9, P.61-

[MIL90] C. Miles & J. McCloskey
"People, the Key to Productivity"
HR Magazine, February 1993, P.42-.

[MON84] M. Moneysmith
"I'm OK - and You're Not"
Tutorial: Software Engineering Project Management, IEEE Computer Society Press, P.346-347]

[MUL] H. Muller, S. Tilley, M. Orgun, B. Corrie & N. Madhavji
"A Reverse Engineering Environment Based on Spatial and Visual Software Interconnection Models"

[NAS92] K. Nash
"Whipping Worn-Out Code into New Shape"
ComputerWorld, Aug.17/92, Vol.26, No.33, P.69-

[NEW90] W. Newsom
"Motivate, Now!"
Personnel Journal, Feb.90, P.50-55

[NIE92] J. Nielsen
"The Usability Engineering Life Cycle"
IEEE Computer, March 1992, P.12-22

[OST89]* E. Ostertag & J. Hendler
"An AI-based Reuse System"
Tech. Rep. CS-TR-2197, UMIACS-TR-89-16, Univ. of Maryland, Dept. of Computer Science, Feb. 1989

[OST92] E. Ostertag, J. Hendler, R. Prieto-Diaz, C. Braun
"Computing Similarity in a Reuse Library System: An AI-Based Approach"
ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 3, July 1992, P.205-228

[OUC81] W. Ouchi
"Theory Z: How American Business Can Meet the Japanese Challenge"
Reading, Mass.: Addison-Wesley, 1981.

[PAR82] M. Parker
"Enterprise Information Analysis: Cost-Benefit Analysis and the Data-Managed System"
IBM Systems Journal, Vol 21, No. 1, 1982, P.108-123

[PAR93] D. Parker
"The Automated Office: a Complement to Business Reengineering"
Working Paper, Feb 1993.

[PER91] A. Percy
"A Plague on 'Open Systems!'"
Gartner Group, Inc., SES Key Issues K-036-1004, Jun.19/91.

[PRE92] R. Pressman
"Software Engineering - A Practitioner's Approach"
3rd Edition, McGraw-Hill, Inc. 1992

[PET82] T. Peters
"In Search of Excellence"
Harper & Row, New York, 1982

[PFA83]* G. Pfaff, editor
"User Interface Management Systems"
Proceedings of the Workshop on User Interface Management Systems, Seeheim, FRG, Nov 1-3, 1983.

[PIT90]* J. Pittman, C. Kitrick, Wavefront Technologies, Inc.
"VUIMS: a Visual User Interface Management System"
Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology, Snowbird, Utah, USA, Oct 3-5, 1990.

[PLI93] N. Pliskin, T. Romm, A. Lee & Y. Weber
"Collaboration Management in DiCE"
The Computer Journal, Vol.36, No.1, 1993, P.87-96

[POT91] J. Potter
"Employee Empowerment: Designing the Infrastructure"
Manager's Magazine, Vol.2, No.3, Fall/Winter 1991

[PRI90]* R. Prieto-Diaz
"Classification of Reusable Modules"
Software Reusability, Vol I: Concepts and Models
T. Biggerstaff & A. Perlis, Eds. ACM Press Frontier Series, Addison-Wesley, Reading, Mass., 1990

[PRI92] T. Prince & D. Kniefel
"What's the Object(ive)?"
ComputerWorld, Oct.5/92, Vol.26, No.40, P.81-

[PUT78] L. Putnam
"A General Empirical Solution to the Macro Software Sizing and Estimating Problem"
IEEE Transactions on Software Engineering, Vol.4, July 1978, P.345-361

[PUT79] L. Putnam & A. Fitzsimmon
"Estimating Software Costs Part I"
Datamation, September 1979

[RAD92] A. Radding
"Pain Pleasure"
ComputerWorld, Sep.14/92, Vol.26, No.37, P.37-

[RAD93] A. Radding & J. Maglitta
"Techno Renaissance"
ComputerWorld, Apr.26/93, Vol.27, No.17, P.67-68

[RAP93] R. Rapaport
"To Build a Winning Team: An Interview with Head Coach Bill Walsh"
Harvard Business Review, January-February 1993, P.110-120

[RAY92] G. Ray
"Software Reuse Not a Panacea"
ComputerWorld, Dec.21/92, Vol.26, No.51, P.47-

[RAY92a] G. Ray
"Windows Builders Eye Promise of C Code Generator"
ComputerWorld, Aug.31/92, Vol.26, No.35, P.85-

[RAY92b] G. Ray
"Reverse-Engineering in C"
ComputerWorld, Aug.17/92, Vol.26, No.33, P.69-

[RAY92c] G. Ray
"Redevelopment Takes Root"
ComputerWorld, Nov.23/92, Vol.26, No.47, P.63-

[REN90] D. Rennie
"Internal Communications: Stakes and Stakeholders"
Manager's Magazine, Vol.2, No.1, Summer '90, P.22-25

[RIC81] G. Richardson & G. Pugh
"Introduction to Systems Dynamics Modeling and DYNAMO"
Cambridge, MA: The M.I.T. Press, 1981.

[RIC88] C. Rich & R. Waters
"The Programmers' Apprentice: A Research Review"
IEEE Computer, November 1988, P.11-

[ROS91] B. Rosser
"To Justify IT Architecture, Go Beyond 'Motherhood'"
Gartner Group, Inc., IS Key Issues, K-140-857, Dec.16/91.

[ROS92] B. Rosser
"What Do You Mean by 'IT Architecture'?"
Gartner Group, Inc., IS Key Issues, K-140-869, Jan.22/92.

[ROY70] W. Royce
"Managing the Development of Large Software Systems"
Tutorial: Software Engineering Project Management, IEEE Computer Society Press

[RUM88] G. Rummler & A. Brache
"Process Management - Managing the White Space on the Organization Chart"
The Rummler-Brache Group, 1988.

[SAM80] P. Samuelson & A. Scott
"Economics"
McGraw-Hill Ryerson, 5th Ed., 1980.

[SAS90] M. Sashkin & R. Williams
"Does Fairness Make a Difference?"
Organizational Dynamics, Vol.19, Autumn 1990, P.56-71

[SCH92] W. Schatz
"What is Reengineering, Anyway?"
Computerworld, Aug.31/92, Vol.26, No.51.

[SCH92] W. Schatz
"Who's Calling, Please?"
ComputerWorld, Nov.23/92, Vol.26, No.47, P.69-

[SCH93] D. Schnitt
"Reengineering the Organization Using Information Technology"
Journal of Systems Management, January 1993, P.14-20

[SHO83] M. Shooman
"Software Engineering - Design Reliability and Management"
McGraw-Hill, Inc., New York, 1983.

[SMI91a] R. Smith
"New Data Administration Dictates of Re-Engineering"
Gartner Group, Inc., SES Strategic Planning SPA-970-854, Nov.26/91.

[SMI92b] R. Smith
"Business Process to Corporate Systems a First Step"
Gartner Group, Inc., IS Key Issues, K-980-837, Sep.30/91.

[SMI92c] R. Smith
"Information Age Triggers Structural Change"
Gartner Group, Inc., IS Key Issues, K-980-867, Jan.22/92.

[SPI76] M. Spier
"Software Malpractice - A Distasteful Experience"
Tutorial on Software Restructuring, R. Arnold, IEEE Computer Society Order Number 680, P.123-129

[STR92]* P. Straub & E. Ostertag
"EDF: A Formalism for Describing and Reusing Software Experience"
International Symposium on Software Reliability Engineering (Austin, Tex., May 17-18, 1991), 106-113

[STU92] P. Stuart
"Fresh Ideas Energize Reward Programs"
Personnel Journal, January 1992, P.102-103

[SUL92]* K. Sullivan & D. Notkin
"Reconciling Environment Integration and Software Evolution"
ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 3, July 1992, P.229-268

[TAY92a] D. Taylor
"Re-Engineering Industries: Eliminate the Middleman"
Gartner Group, Inc., IES Key Issues K-912-552, Nov.9/92.

[TAY92b] D. Taylor
"Re-Engineering Industries: The Middleman Fights Back"
Gartner Group, Inc., IES Key Issues K-912-553, Nov.9/92.

[TER91] R. Terdiman
"Business Issue No. 1: Process Re-Engineering"
Gartner Group, Inc., IS Key Issues, K-980-848, Nov.26/91.

[TER92] R. Terdiman
"BPR Case Study: Fairfax County, Virginia"
Gartner Group, Inc., IS Events, E-955-906, Jul.28/92.

[TER93] D. Terpstra & E. Rozell
"The Relationship of Staffing Practices to Organizational Level Measures of Performance"
Personnel Psychology - A Journal of Applied Research, Vol.46, No.1, Spring 1993, P.27-48

[TER93a] R. Terdiman and M. Light
"BPR: Critical Success Factors"
Gartner Reports, IS Key Issues, K-955-949/951, Jan. 22/93

[TER93b] R. Terdiman and M. Light
"BPR: A Three-Part Combination"
Gartner Reports, IS Key Issues, K-955-945, Jan. 22/93

[THA84] A. Thadhani
"Factors Affecting Programmer Productivity During Application Development"
IBM Systems Journal, Vol.23, No.1, 1984, P.19-35

[TIL92a] S. Tilley
"Management Decision Support Through Reverse Engineering Technology"
Proccedings of CASCON '92, Toronot, Ont., November 1992.

[TIL92b] S. Tilley, H. Muller & M. Orgun
"Documenting Software Systems with Views"
SIGDOC '92

[TRA88] W. Tracz
"Confession of a Used Program Salesman - Reuselss Software"
IEEE Computer, December 1988, P.75

[TRA91] W. Tracz
"Confessions of a Used Program Salesman - Intellectual Property Law ???"
IEEE Computer, April 1991, P.112

[TYS92] S. Tyszberowicz & A. Yehudai
"OBSERV - A Prototyping Language and Environment"
ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 3, July 1992, P.269-309

[UNI91]* UNIX System Laboratories, Inc.
"Open Look Graphical User Interface - User's Guide"
Prentice Hall, 1991.

[VAN88] G. VanderBurg and E. Bush
"A CASE for Existing Systems"
Proceedings "Chaos into Order", CIPS Edmonton '88, Nov. 15-17, 1988, Edmonton, Alta, P.137-146

[VAR92a] Various Authors
"Business Process Re-Engineering: Key Issues"
Gartner Group, Inc., IS Vol.-008-006, Feb.12/92.

[VAR92b] Various Authors
"Banking on Client/Server: A Case in Point"
Gartner Group, Inc., IS Vol.-008-006, Feb.12/92.

[VAR92c] Various Authors
"Lester Thurow on the Evolution of High Technology"
Gartner Group, Inc., IS Vol.-008-004, Jan.29/92.

[VAS82]* Y. Vassiliou, editor
"Human Factors and Interactive Computer Systems"
Proceedings of the NYU Symposium on User Interfaces, New York, May 26-28, 1982.

[VIN93] H. Vin, M. Chen & T. Barzilai
"Presumed versus Actual Organizational Culture - Managerial Implications for Implementation of Information Systems"
The Computer Journal, Vol.36, No.2, 1993, P.143-152

[VIT92] J. Vitello, compiler
"Being Too Perfect Can Hurt an IS Professional"
ComputerWorld, Aug.10/92, Vol.26, No.32, P.81

[WAG90] W. Wagel
"Make Their Day - The NonCash Way!"
Personnel, May 1990, P.41-44

[WAL87]* R. Wallace, E. Stockenberg, R. Charette
"A Unified Methodology for Developing Systems"
McGraw-Hill, 1987.

[WEI71] G. Weinberg & M. Fisher
"The Psychology of Computer Programming"
Litton Educational Publishing, Inc., New York, 1971.

[WEI79] K. Weick
"The Social Psychology of Organization"
2nd Ed., Reading, MA: Addison-Wesley Publishing, 1979.

[WEI83] G. Weinberg
"Kill That Code"
InfoSystems, August 1983.

[WEI86]* M. Weiser & B. Shneiderman
"Human Factors of Software Design and Development"
Tutorial on Software Restructuring, R. Arnold, IEEE Computer Society Order Number 680, P67-81.

[WEL90] R. Welford
"Worker Motivation, Life Time Employment and Codetermination: Lessons from Japan and West Germany for Productivity Growth"
Contempory Review, Vol.257, No.1496, Sep.90, P.129-132

[WES92] M. West
"Information Engineering With Objects: A Viable Strategy"
Gartner Group, Inc., ADM Strategic Planning, SPA-700-740, Jul.21/92.

[WRI93] T. Wright & D. Bonnett
"Role of Employee Coping and Performance in Voluntary Employee Withdrawal: A Research Refinement and Elaboration"
Journal of Management, Vol.19, No.1, Spring 1993, P.147-161

[YAK90] B. Yakemovic and J. Conklin
"Report on a Development Project Use of an Issue-Based Information System"
Proceedings of the Conference on Computer-Supported Cooperative Work, Oct. 7-10, 1990, Los Angeles, CA, Sponsored by ACM SIGCHI & SIGOIS, P.105-118

[ZEL92] M. Zelkowitz
"Are Software Engineering Process Standards Really Necessary?"
IEEE Computer, November 1992, P.82-84