Exceptions and Exception Handling in Computerized Information Processes

DIANE M. STRONG
Boston University
and
STEVEN M. MILLER
Fujitsu Network Transmission Systems

Exceptions, situations that cannot be correctly processed by computer systems, occur frequently in computer-based information processes. Five perspectives on exceptions provide insights into why exceptions occur and how they might be eliminated or more efficiently handled. We investigate these perspectives using an in-depth study of an operating information process that has frequent exceptions. Our results support the use of a total quality management (TQM) approach of eliminating exceptions for some exceptions, in particular, those caused by computer systems that are poor matches to organizational processes. However, some exceptions are explained better by a political system perspective of conflicting goals between subunits. For these exceptions and several other types, designing an integrated human-computer process will provide better performance than will eliminating exceptions and moving toward an entirely automated process.

Categories and Subject Descriptors: I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*industrial automation*; *office automation*; J.1 [**Computer Applications**]: Administrative Data Processing—*business*; K.4.3 [**Computers and Society**]: Organizational Impacts; K.6.2 [**Management of Computing and Information Systems**]: Installation Management—*performance and usage measurement*; K.6.4 [**Management of Computing and Information Systems**]: System Management—*quality assurance*

General Terms: Design, Management, Performance

Additional Key Words and Phrases: Exceptions, exception handling, process design, Total Quality Management (TQM)

1. INTRODUCTION

Despite the fact that computers are touted as labor saving and time saving, exceptions occur frequently in computerized information processes [Gasser 1986; Suchman 1983]. Exceptions are cases that cannot be correctly processed

This research was funded by the anonymous field site company.
Authors' addresses: D. M. Strong, Boston University, School of Management, 704 Commonwealth Avenue, Boston, MA 02215; email: dstrong@acs.bu.edu; S. M. Miller, Fujitsu Network Transmission Systems, 2801 Telecomm Parkway, Richardson, TX 75082.

by computer systems alone, and thus require manual interventions to produce outputs that meet organizational goals. These manual interventions result in reduced productivity and increased processing time. Furthermore the exception-handling process itself can introduce new errors and thereby reduce the quality of process outputs [Kling and Iacono 1984a]. Because of these adverse performance effects, managers attempt to eliminate exceptions by improving the capabilities of computer systems. Vendors sell systems based on this same reasoning. However, in spite of these efforts, exceptions are common in computerized information processes. For example:

> Staff in one company routinely corrected inventory information before using it because the computer-based data was not accurate enough for decision making [Kling and Iacono 1984b].
>
> Engineers at another company learned to enter "incorrect" parameters so they could get correct results from a computer system [Gasser 1986].
>
> Order processors at our field site routinely corrected inappropriate plant assignments generated by a computer system before the system forwarded the information to that plant.

In this article, we investigate the causes of exceptions in computer-based information processes and the usefulness of routine procedures for handling these exceptions. We focus on routine, operational-level information processes, e.g., accounts receivable and payable, inventory control, order fulfillment. These processes are typically highly computerized, and yet they still require significant human resources to accomplish their goal adequately.

We start from five alternative perspectives on exceptions that provide a basis for considering whether and how exceptions can be eliminated. One perspective views exceptions as infrequent, nonrepetitive events about which little can be forecast. Two perspectives are variations on the theme that exceptions in information processes are "bad"; they are signals of poor process quality that can and should be eliminated to improve performance. The other two focus on the persistence of frequent exceptions ranging from understanding why exceptions are difficult or impossible to eliminate to why exceptions are a useful and important part of process capabilities.

We investigate these perspectives in light of empirical data from a computer-based information process supporting order fulfillment in a large organization. Our findings indicate that performance improvement depends on distinguishing between exceptions that can and should be eliminated and exceptions that are key to effective and flexible information processes. Understanding the causes of exceptions provides the basis for (1) reducing or eliminating some exception types and (2) more astutely handling exception types that are important to achieving process goals.

2. PERSPECTIVES ON EXCEPTIONS

For routine organizational processes, computer systems serve to structure, rationalize, and routinize work [Markus 1983]. The measure of performance of these computer-based information processes has traditionally been opera-

tion without manual intervention [Bainbridge 1987]. That is, if we captured correctly an information process within a computer system, the computer system would repeatedly and correctly perform that process. The computer system was either operating correctly, or it had errors requiring manual intervention. These errors could be operation errors or process design errors, but they were all *errors* that represented less than perfect performance of the computer-based process [Kling 1980].

However, from years of experience with real computer-based systems in real organizations, we know that this binary view of the world as correct or in error is too narrow [Kling 1980]. Manual interventions in routine computer-based processes occur frequently [Gasser 1986; Rasmussen et al. 1987; Sirbu et al. 1984; Suchman 1983], and these interventions are not necessarily caused by errors [Gasser 1986]. We take a broader view in this article and consider the purposes of manual interventions in computer-based processes, which we call exception handling, and their contribution to the performance of the entire information process.

We define exceptions in computer-based information processes as cases that computer systems cannot process correctly without manual intervention, a definition broader than "errors." This definition of exceptions covers those generated by incomplete and erroneous information in inputs and outputs, requests to deviate from standard procedures, and situations that computer-based systems were never designed to handle. It is consistent with a dictionary definition of "exception" as "a case to which a rule does not apply"[1] in the sense that, by employing a thorough systems analysis of a routine information process, all applicable rules are embedded in computer systems. That is, the decisions made in routine processes are commonly assumed to be programmable and suitable for computer-based systems [March and Simon 1958]. Cases computer systems do not process correctly are exceptions to the decision rules in these systems. This definition excludes manual processing that is not an intervention to cover cases that computer system rules do not cover; that is, activities such as routine record keeping, paper movement, printing and distributing reports, gathering input information, workload planning, and training are not considered to be exceptions.

We take this broad view of exceptions because the presence of manual interventions in computer-based systems has been always viewed as less than perfect performance. Manual interventions, necessarily, have implications for process performance and thus for organizational performance. However, because exceptions are not necessarily errors, the answer to less than perfect performance is not necessarily to eliminate exceptions.

In the real world, people understand that computer systems are not always "correct"; there are exceptions requiring manual intervention. Computer-based systems are typically surrounded by manual exception-handling procedures, many of which are routine procedures developed as responses to routine exceptions. For example, Gasser [1986, p. 212] discusses common

[1] *Websters' Ninth New Collegiate Dictionary.*

situations in which "computing is misfit to the work it is intended to support" and describes three strategies for accommodating computing misfit—fitting, augmenting, working around—that are often critical to obtaining satisfactory performance using computer systems. Computer system performance cannot be evaluated without considering the performance of these surrounding exception-handling routines. However, people have not paid much theoretical, analytic, or managerial attention to these procedures. As a result, exception-handling procedures and computer system work-arounds were never explicitly designed and were often more inefficient than necessary.

In the above discussion, there are two conflicting, yet widely held, views of the nature of (what we are calling) exceptions in routine computer-based processes. One view is that exceptions are a normal part of organizational processes. Few organizational researchers or practicing managers would argue that all routine operational decisions are programmable. By their nature, organizational processes, even highly routinized ones, involve some decision making, problem solving, and information processing requiring capabilities and judgment of people [Suchman 1983].

Although few would argue against the view of exceptions as natural to organizational processes, much current research and organizational practice assumes an alternative view that routine decisions and processes are programmable and should be embedded in computer systems for efficient operation of processes. This view has a long research tradition. It is evident in the research of Simon and associates (e.g., March and Simon [1958] and Simon [1977]) and continues with expert system researchers, (e.g., Goldstein and Storey [1991], Lenat et al. [1990], and Storey and Goldstein [1993]) who are working to add judgment, common sense, and other human abilities to computer systems. Expert system researchers often view systems as inadequate until they are capable of replacing human decision makers (e.g., Goldstein and Storey [1991] and Storey and Goldstein [1993]) although other researchers are working toward design support systems that explicitly incorporate human decision makers (e.g., Cohen and May [1992] and Cohen and Strong [1991]). Computing resources in general and expert systems in particular are viewed as increasing the information-processing capabilities of firms [Galbraith 1973; 1977; Sviokla 1990].

This automated-systems view is further supported by current management practices of process reengineering, which seeks to rationalize and computerize information processes [Davenport 1993; Hammer 1990], and total quality management, which seeks to find and eliminate sources of variation in organizational processes [Deming 1986; Juran 1989]. The focus of these research and management efforts is performance improvement by eliminating exceptions, rather than improving the performance of routines for handling exceptions.

To explore further these two general views of exceptions and what should be done about exceptions in routine organizational processes, we present five perspectives on exceptions, which are shown in Figure 1. One perspective views exceptions as infrequent random events (row 1 in Figure 1). Two perspectives focus on exceptions as errors to be eliminated (errors at-

Underlying Assumption	Perspectives on Causes of Exceptions	Perspectives on Solutions to Exceptions	Solution Approach
Exceptions are unpredictable.	1. Random Event	None	None
Exceptions are errors, indicators of process problems.	2. Errors (from operations, design, and dynamic organizations)	4. Total Quality Management (TQM)	Eliminate causes of exceptions.
Exceptions are "normal," part of process flexibility.	3. Political System	5. Human-Computer System	Efficiently detect and handle exceptions.

Fig. 1. Perspectives on exceptions.

tributable to various process problems and total quality management); these are the typical perspectives adopted by managers and information systems researchers (see row 2 in Figure 1). Our interest in this article is to contrast this view with the view in the third row of Figure 1, exceptions as a normal part of organizational processes. The two perspectives in row 3 in Figure 1 (political system and human-computer system) seek to understand why exceptions persist in spite of attempts to eliminate them and consider how exception-handling procedures could be more efficiently performed as part of normal process operations. We discuss first the three perspectives on the causes of exceptions (random event, error, and political system) followed by the two perspectives on solutions to exceptions, i.e., eliminating them (total quality management) or more efficiently handling those that persist (human-computer system).

2.1 Random-Event Perspective

The word "exception" connotes typically rare and infrequent events. The random-event perspective captures this connotation of exceptions. According to this perspective, exceptions are low-probability events that are unexpected, nonrepetitive, and infrequent. They include both random errors during normal processing and such events as fires, floods, and computer system downtime that could disrupt processing.

This perspective is commonly assumed by managers and researchers; people assume that computer systems will work correctly most of the time and that exceptions will occur only rarely. However, a random-event perspective is not supported by research studies. Exceptions occur frequently in information processes [Gasser 1986; Sasso et al. 1987; Sirbu et al. 1984;

Suchman 1983]. Some occur so frequently that the response to them becomes routinized [Gasser 1986; Sirbu et al. 1984]. The frequency of exceptions is a major difficulty in systematically analyzing office operations [Sasso et al. 1987].

The random-event perspective is included for completeness. Truly random events cannot be eliminated, nor can efficient routine procedures be developed to handle them. Thus, they will not be discussed further.

2.2 Error Perspective

Exceptions may be caused by errors—operation errors, process design errors, or errors due to dynamic organizations. Mistakes made by people are generally thought of as operations errors, whereas mistakes made by physical systems such as computer systems are typically thought of as design errors [Rasmussen et al. 1987]. That is, people can make mistakes, but computer systems perform as they were designed to perform; so their "mistakes" are classified as design errors or as random events, e.g., downtime.

Operation Errors. Operation errors include mistakes in processing (e.g., promising delivery when there is no inventory) and mistakes in inputs to the process (e.g., orders for nonexistent products). In highly computerized processes, operations errors in the form of mistakes made by people are rare because people are not doing the processing. Operations errors can be common in manual interventions because exception handling may introduce new errors into the process.

Design Errors. Managers and researchers may interpret the presence of exceptions as evidence of poor process design; that is, if the information process, especially the computer systems, had been designed and implemented correctly, then there would be only random-event exceptions. Research on the difficulty of understanding organizational processes provides some support for this interpretation of exceptions [Anderson 1980; Cohen and Bacdayan 1994; Ericsson and Simon 1984; March and Simon 1958; Nisbett and Wilson 1977; Stinchcombe 1990; Whitten et al. 1989]. Even if an accurate representation of an existing process is available, (1) the process of design is generally complex and not well understood [Simon 1981], (2) the knowledgeable design of organizational routines and information processes is especially difficult [Cohen and Bacdayan 1994; Galbraith 1973; 1977], and (3) the result of applying information technology in organizations is not predictable [Markus and Robey 1988]. In addition, many operational processes were not explicitly designed but were gradually grown [Hammer 1990].

Dynamic Organizations. Exceptions caused by organizational changes are a variation on design errors. Organizational procedures and goals, even for routine processes, evolve over time [Nelson and Winter 1982]. A static process captured by systems analysis and embedded in computers will not accurately represent an organization for long. Over time, the mismatch between the routines embedded in computer systems and organizational decision rules may gradually increase, resulting in more exceptions. These exceptions repre-

sent cases that computer systems were never designed to process because these cases did not exist when the computer systems were developed.

If computer systems are not kept up to date with organizational decision rules, people will gradually develop routines for recognizing and handling the new cases that computer systems cannot process correctly. This gradual development of exception detection and handling is consistent with an observed characteristic of organizational routines as gradual learning by multiple actors over time [Cohen and Bacdayan 1994]. Exception-handling work is likely to increase over time as the mismatch between the computerized system and the organization gradually increases.

2.3 Political System Perspective

A political system perspective (e.g., Kling and Iacono [1984a] and Markus [1983]) explains the persistence of some exceptions, especially in information processes that cross organizational boundaries, e.g., order fulfillment starts in sales and continues into manufacturing. Different subunits, such as sales and manufacturing, are likely to have different and possibly conflicting goals, which may be captured in computer systems to varying degrees. For example, the unit with the most political power may be able to implement its solution [Kling and Iacono 1984a]. In general, computer systems developed in the context of conflicting goals are unlikely to have met the goals of all subunits [Franz and Robey 1984; Kling 1980].

Goal conflict is likely to result in exceptions. That is, the goals of less powerful subunits still exist and may need to be addressed even if these subunits failed to achieve their goals at the time of computer systems development. Exception handling then serves the role of meeting, to some degree, the needs of these less powerful subunits. Conflicting subunit goals make it difficult to eliminate these exceptions since there may not be a solution that is satisfactory, let alone optimal, for all units involved.

2.4 Total Quality Management (TQM) Perspective

A Total Quality Management (TQM) perspective is a "solution" perspective rather than a "causes" perspective; it focuses on what to do about exceptions rather than positing an underlying cause for exceptions. A TQM perspective assumes that exceptions are systematic errors that should be eliminated. These errors are eliminated by (1) finding the root causes of the most frequent or costly exceptions and then (2) eliminating these root causes [Case 1987; Deming 1986; Fiegenbaum 1991; Ishikawa 1985; Juran 1989]. The repeated application of these steps is the continuous-improvement aspect of TQM. Continuous improvement differs from process redesign, which attempts more radical improvements [Davenport 1993].

As a result of a TQM approach, work is done correctly the first time rather than by inspecting and reworking to achieve quality [Case 1987; Deming 1986; Fiegenbaum 1991; Ishikawa 1985; Juran 1989]. The goal of a TQM approach is process performance in which the only problems are truly ran-

dom events or errors. All systematic errors have been identified and eliminated.

2.5 Human-Computer System Perspective

A human-computer system perspective focuses on the employment of people and computer systems to form an integrated human-computer process. Like a TQM perspective, a human-computer system perspective is a "solution" perspective. According to this perspective, both people and computer systems add value to the process [Simon 1977; Strong 1989]. Computer systems store and process information and report on problems. People monitor the operation of the process and provide process flexibility that is difficult to achieve with computer systems.

In this perspective, exceptions are legitimate special cases. The goal is not to computerize the entire process, but to employ both human and computer resources appropriately. Exceptions that are a key part of the flexible operation of the process should be efficiently handled rather than eliminated. In this perspective, inefficiencies occur when the tasks of people are not adequately integrated with, and supported by, computer systems and vice versa.

One aspect of this perspective is to evaluate the costs and benefits of using computer systems or people to perform tasks within routine processes. For example, it may not be cost effective to capture all possible cases in computer systems. Economic choices are made between using people or computer systems for handling work based on the frequency of exceptions, the difficulty of capturing and maintaining computerized versions, and the difficulty of handling them manually. Thus, exceptions represent sensible economic decisions rather than signals of process problems.

3. METHOD

Our research goal was to develop understanding about exceptions and derive managerial recommendations for treating exceptions. To accomplish this goal, we investigated the applicability of the alternative perspectives on exceptions using an in-depth study of an operating information process in one organization. Although a single-site study necessarily limited the generalizability of our findings, the level of detail available in such a study provided evidence for the perspectives and examples to illustrate their applicability [Benbasat et al. 1987].

3.1 Field Site

The field site was a Fortune 100 firm[2] that manufactures large, expensive electronics systems that were sold to other firms for use in information-processing applications. It was an international firm with sales and manufacturing facilities in many countries. The firm had a general reputation for engineering excellence. Since the lifetime of its products was short, it was continually designing, manufacturing, and selling new products.

[2]The firm has requested anonymity.

We studied the information process that supported order fulfillment for build-to-order manufacturing in the United States. This process was the responsibility of the manufacturing organization and served as one of manufacturing's primary interfaces with the sales organization. It was organized by product groups and was physically located in the same, or nearby, buildings as associated product manufacturing.

Although some characteristics of this process are unique to this firm, order fulfillment is a common process in manufacturing and service organizations. Since order processing provides manufacturing with information needed for production, its successful operation is critical to the financial well-being of manufacturing organizations. One reorganization of order processing at our field site led to orders not being processed and a significant decline in revenue. Other firms have had similar experiences. Thus, our field site is deliberately and carefully improving the quality of the information from its process and the efficiency of the process.

3.2 Sources of Data

The object of our study is a process rather than people or organizational units. To develop a detailed understanding of this process, we used expert sources (i.e., key informants) rather than a representative sample of people involved in the process. The informants included one manager, one supervisor, two staff specialists who had previously studied the process, and two expert order processors.

Archival records about the operation of the process, including two previous studies and three reports, were available. The previous studies documented the work and exceptions in this process. The three reports included the following data: summary of processing times for approximately 1,000 orders processed by three order processors during a six-month period, the exceptions found in these orders, and processing details for key activities within the process.

3.3 Data Collection

Collecting data about an operating process is a discovery process necessitating an iterative collection procedure. The two previous studies served as a starting point for understanding the process; however we recollected all these data by interviews and work observation. Informants were interviewed several times until their explanations were sufficiently detailed and verified. Expert order processors demonstrated the process by doing walk-throughs of the process with sample orders of differing complexity. We also observed the process for eight working days. During this observation, we recorded the activities performed, the orders worked on, and the information inputs and outputs.

3.4 Analysis

The interview data were analyzed and summarized in the form of process flow diagrams. These diagrams were iteratively developed, refined, and veri-

fied with process experts. From the interview data, we compiled a list of the major decisions made during order processing, what information was needed to make the decisions, and how this information was acquired and used. The decision-making data were verified using follow-on interviews, the order walk-through data, and the work observation data.

The analysis of the work observation data used the global modeling method from protocol analysis [Todd and Benbasat 1987], which involves coding and then flowcharting. The data were first transformed from the view of order processors performing their daily activities to the view of orders flowing through a process by coding[3] the 602 observed activities. The coded work observation data were than summarized in the form of a flow diagram that was compared to the flow diagram from the interview data.

The interview data, work observation data, and archival records were analyzed to determine the major exceptions occurring during the process, where major exceptions were defined to be exceptions that occurred in more than 15% of the orders or that took more than five minutes to handle per order. Our goal was to determine the exceptions for which routinized detection and handling procedures were likely. We also checked that the major exceptions caused nontrivial manufacturing or customer disruptions if they were not caught and fixed.

For each major exception, the procedures for detecting and handling that exception were compiled from the interview and work observation data. Each *detection procedure* was described as a decision about whether or not[4] an exception exists. Each *exception-handling procedure* was described in terms of any decisions made followed by the actions taken. For each decision made during exception detection and handling, the following are listed:

(1) the decision made,
(2) the information required to make the decision,
(3) the source of this information,
(4) the method of acquiring the information from its source,
(5) how the information was used to make the decision, and
(6) for exception-handling procedures, the actions taken.

This structure captures the decision-making and information-processing nature of exception detection and handling routines and provides some indication of the skills, knowledge, and discretion used when making these decisions.

[3]Coding was done in four passes: the first pass classified activities as part of the process being studied or other; the second pass classified activities into major processing groups; the third pass classified activities within the groups; and the fourth pass cross-referenced activities for the same orders. All activities were coded by the first author. A sample of activities was coded by an independent coder yielding 95% agreement for the first pass and 75% agreement for the second pass.

[4]Although the existence of an exception may form a continuum, the purpose of detection is to decide whether the information is good enough, i.e., a satisficing criterion [Simon 1981], or the information should be further processed by exception-handling activities.

Inefficiencies were found by comparing the procedures for steps that were easy to perform in some procedures, but difficult to perform in others. The focus of this analysis was on the availability of needed information, computer system support, and the knowledge, experience, and expertise required of order processors.

4. FINDINGS

4.1 Overview of Order Processing

The firm processed approximately 100,000 orders each year, each containing approximately 250 pieces of information. Inputs to the information process were customer orders collected by the sales organization. Outputs were customer orders with all the information needed by manufacturing to build the product. The major tasks in the process were: adding information needed by manufacturing, including schedule date, engineering specifications, and build sites (called sources). This was primarily a computerized process, i.e., computer systems produced this additional information. The process was intended to work semiautomatically with only limited intervention from people. However, significant human resources were required during the process. Figure 2 shows this process. At the process starting point, (1) the order has been entered into a computer system, (2) basic order verification has been completed (which means that the firm has accepted the order), and (3) the order has been transferred to the scheduling computer system. All these actions were the responsibility of the sales organization.

Next, the computer system assigned sources (production plants) to each line item in the order and assigned a scheduled ship date to the order. Sourcing was done by a table lookup of each component ordered (each line item was one type of component) to find the plant that produced that component. The scheduled ship date was the last day of the month in which all the schedulable (major) components could be produced according to a previously developed master production schedule stored in the computer system. The computer system then sent the order to order processors in manufacturing.

One hundred employees, called order processors, checked for exceptions in orders, performed exception handling, and, generally, ensured that orders moved through the process in a timely fashion. Order processors were organized into groups by product type: large systems, medium-sized systems, small systems, and special systems. Total processing time for orders within this process ranged from one day to several weeks. Approximately 25% of the long processing times were directly attributable to exception handling.

The basic process shown in Figure 2 has remained essentially the same for at least a 10-year period. However, changes did occur in the organizational decision rules and computer systems used for some steps. During this study, one expert system that produced product configurations was part of the process. A second expert system to make sourcing (build site) decisions was in

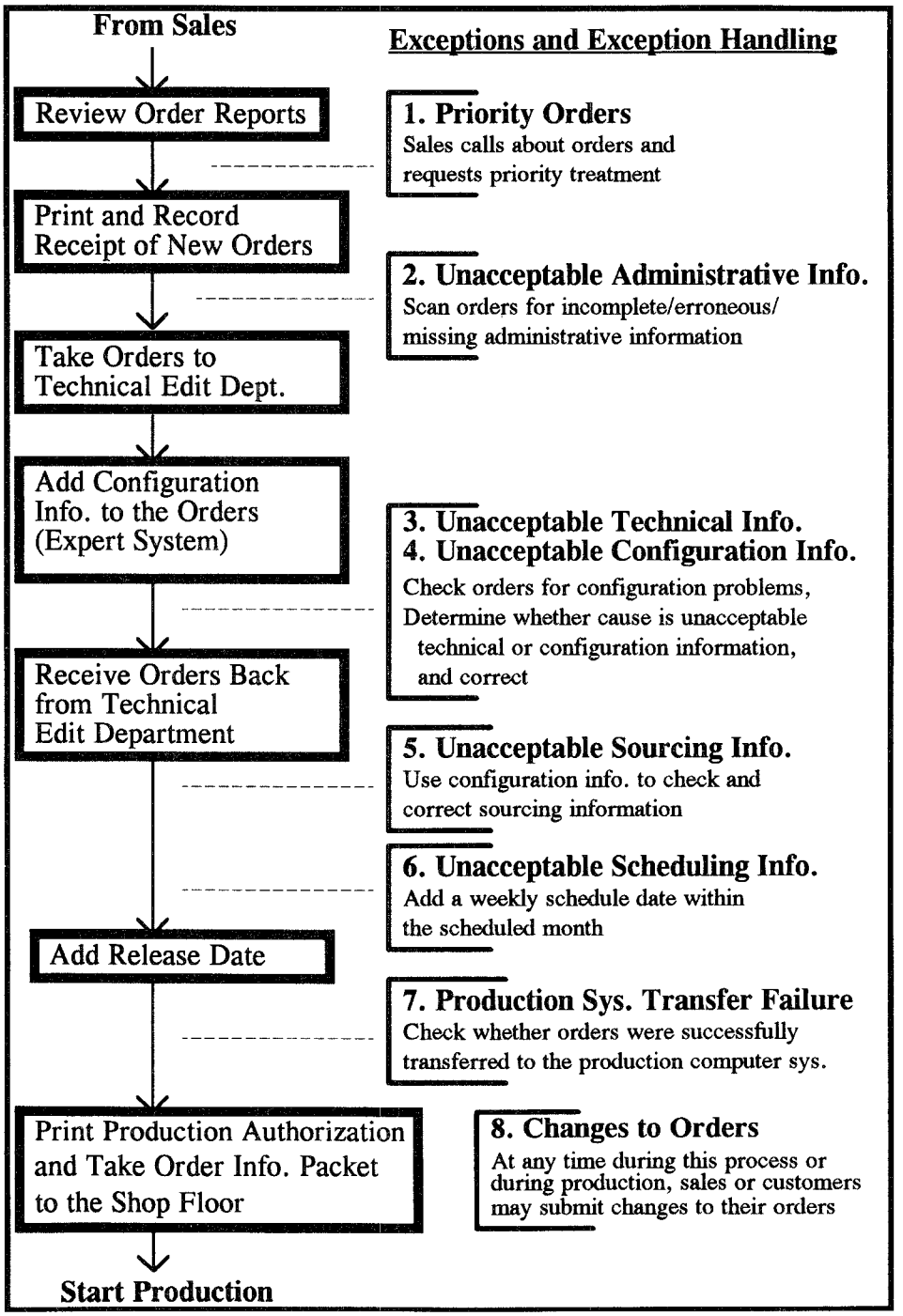

Fig. 2. Order fulfillment process flow.

prototype form, and a third expert system for making scheduling decisions was being developed. The configuration expert system replaced a manual decision-making process. The other two expert systems were replacing or augmenting two conventional computer systems. The sourcing and scheduling computer systems were being upgraded because they no longer were good matches to the organizational process.

4.2 Exceptions

Eight major types of exceptions, listed in Figure 3 with their frequency, occur during this process. Five of these are caused by unacceptable information: unacceptable administrative, technical, configuration, sourcing, and scheduling information. The first two refer to information contained in orders when they arrive from sales. Administrative information consists of order information needed to process the order other than the components ordered, e.g., address, credit, transportation information. Technical information is the components ordered with their quantities. The last three are information added to orders during this process. The remaning three major exceptions are caused by changes to orders, priority orders, and problems in transferring orders between computer systems.

Three of the eight major exceptions had relatively low frequency (1 or 2% of orders); however, exceptions in only 1% of 100,000 orders still require significant exception-handling resources. The other five ranged in frequency from 10% to 100% of orders. Thus, all orders require some form of manual intervention.

4.3 Exception Handling

A walk-through of a medium-complexity order illustrates the roles of computer systems and people in this process and the routine procedures for detecting and handling the eight major exceptions. Figure 2 indicates where these exceptions occur in the process.

The order first comes to the attention of the appropriate order processor via a telephone call from the salesperson to ask for priority treatment of an order that is on its way. (The standard procedure is that order processors find new orders by checking the computer system for new orders for their product(s). However, salespeople call commonly to say an order is coming and perhaps to ask for priority treatment.) The order processor questions the salesperson about the need for priority treatment and makes a judgment based on customer, dollar volume of the order, and previous experience with this salesperson. Although official company policy is that all customers are treated equally and orders are processed on a first-come, first-serve basis, there is an undocumented list of important customers that the order processor knows from experience.

The order processor next accesses the computer system and prints, at his local printer, a copy of the order. He records the order in his personal tracking system, scans the header information for problems, and jots down

Type of Exception	Decision to Make	Orders with the Exception	Applicable Perspective on Exception Cause
1 Priority Orders	Should this order be given priority?	10 %	Error (Design) Political System
2 Unacceptable Administrative Information	Is the administrative information (address, credit, etc.) acceptable?	2 %	Error (Operations) Political System
3 Unacceptable Technical Information	Is the technical information (the components ordered) acceptable?	10 %	Error (Operations) Political System
4 Unacceptable Configuration Information	Is the configuration information acceptable?	1 %	Error (Design) Error (Dynamic Org.)
5 Unacceptable Sourcing Information	Is each order line item sourced to the correct plant?	25 - 50 %	Error (Dynamic Org.)
6 Unacceptable Scheduling Information	Which week in the scheduled month should be assigned for production?	100 %	Error (Dynamic Org.)
7 Production System Transfer Failure	Was the order successfully transferred to the production computer system?	1 %	Error (Operations)
8 Changes to Orders	Have changes to an order been submitted?	? %	Error (Design) Political System

Fig. 3. Exceptions in the order fulfillment process.

these problems (e.g., the order does not yet have credit approval, or the ship-to-address is missing). He does not attempt to resolve these problems because other people should be attending to them (e.g., the credit department should be completing the credit check). Then, he takes the order, along with any others that arrived in the last hour or so, to the technical edit department, which will attach configuration information (diagrams) to the order.

On the computer system, the order has already been configured by an expert system. The technical editor checks the computer system output for

problems, finding one. The computer system indicates that one component cannot be configured into the product. The technical editor makes an assessment about whether this component is supposed to be an extra spare part (in which case the order is acceptable), whether there is a mistake in specifying the component, or whether the expert system was in error. To resolve the problem the technical editor calls the order processor who may need to call the salesperson who may need to contact the customer, all of which may cause a significant processing delay. When the problem is corrected, the computer system prints the configuration diagrams for the technical editor who returns the order with the diagrams to the order processor.

Next, the order processor checks that the sources assigned earlier by the computer system are compatible with the configuration diagrams. (Components configured into the same cabinet must be produced at the same source, but the sourcing system does not consider this constraint.) The order processor reads the configuration diagram to find components in the same cabinet and then scans each line item of the order in the computer system. The order processor manually overrides incorrect sources in the computer system.

The order processor next checks his microproduction schedule for the month and selects a week within the month for which the order is scheduled. He then assigns this date as the release date in the computer system. After a release date is assigned, the computer system, in an overnight batch process, transfers the order to the production computer system.

The next morning the order processor prints a list of the previous night's activities (or picks up a printed report) and checks whether the computer system transfer was successful. If not, he investigates the cause, which is usually either missing information in the BOM (bill of materials) on the production system or an old version of the order already on the production system, fixes the problem (probably by talking with the production computer system personnel), and rereleases the order. The following morning this checking procedure is repeated. When the transfer between computer systems is completed successfully, production and assembly of the ordered components can begin.

At any time during this process or after production begins, the customer or the salesperson may initiate a change to the order. This requires special problem solving by the order processor to determine what processing, if any, must be revised.

Overall, the exception-handling procedures in this process serve their purpose. Most exceptions in this process are caught and fixed during the process. Approximately 92% of orders complete the process with no exceptions remaining. Eight percent of orders have an uncaught exception at the end of order processing. These exceptions are caught in manufacturing or at the customer site. Considering that 100% of the orders had some form of exception during the process, 92% is good performance. However, the uncaught exceptions represent 8,000 orders—hardly a signal of excellent performance. Most of these exceptions are caught in manufacturing, resulting in some form of disruption or resource expenditure (e.g., when a plant assigned

to produce a portion of the order is unable to produce that part, it must be reassigned to a different plant). Exceptions caught at the customer site range from products delivered later than promised to products that do not function because they were incorrectly configured.

4.4 Efficiency of Exception-Handling Procedures

Our efficiency analysis found three general inefficiencies in exception-handling procedures. These inefficiencies required the use of more resources (usually the time of people) than other exception-handling procedures. More exception-handling resources than usual were required when:

(1) detecting exceptions required 100% manual inspection,
(2) information needed for decisions was not readily available, and
(3) controls in computer systems were too restrictive.

Detecting Exceptions. 100% inspection of orders to find exceptions was needed when computerized support was not provided. For example, at the time of our study, the sourcing computer system no longer produced correct sources according to current company rules. Order processors had to scan every line item of every order to check for incorrect sources. Although, in practice, this was relatively fast because the order processors knew the types of mistakes the system made, methods more efficient than 100% inspection could improve performance.

Difficulty in Acquiring Information. Acquiring the information to make decisions about exceptions and their handling could be time consuming. For example, priority treatment decisions required the use of undocumented information acquired through experience. Novices acquired the needed information by searching within the organization (e.g., they asked someone). In general, novices spent more time searching for solutions to resolve complex decision cases. As a result, experts processed 1.5 times as many orders as novices in all time periods. Additionally, some order information must be acquired from salespeople or customers after the order has been accepted for production, which can cause significant delays in processing the order.

Controls in Computer Systems. Controls in computer systems increased the difficulty of making routinely requested changes to orders. For example, one computer system has controls to prohibit changes to orders after they have been scheduled for production in the current month. Since orders are frequently changed within this time frame, exception handling for change requests is unreasonably difficult and time consuming. For example, computer system controls require resubmitting some orders revised by sales as if they were entirely new submissions, which resulted in possibly unnecessary rework. Because the stated policy and the actual operating policy differ, excessive exception-handling resources were consumed when the computer

system was designed to implement the stated policy of not allowing changes within 30 days of delivery.

5. DISCUSSION

5.1 Error Perspective

Operation Errors. Three of the major exceptions, administrative information, technical information, and transfer to the production system, can be partially understood as operations errors. For administrative and technical information, operations errors can occur in gathering the information from customers and in entering it into a computer system. Salespeople often lack the knowledge to specify correctly the components needed in a complex product; this results in incorrect technical information. However, many administrative and technical information exceptions are explained better by the political perspective. For transfer exceptions, erroneous information in the production computer system (e.g., old bill of materials information) may cause the production system to refuse to accept orders transferred to it. Transfer exceptions may occur also when the communication link between the computer systems fails (a rare, random event). Additionally, operations errors occur as people perform exception-handling routines. For example, order processors may fail to detect incorrect sources (production plants), or they may correct an incorrect source with another incorrect source.

Design Errors. Design assumptions in computer systems contribute to difficulties and inefficiencies in performing exception handling. For example, computer systems were designed to implement the official policy of complete orders, rather than the actual operating procedures of starting production on incomplete orders. Computer system controls that made it difficult to change orders after production started served only to complicate the process of completing order information. A similar case occurs for priority orders. The computer systems provided no support for priority orders because the system design assumed a first-come, first-serve policy for order fulfillment. Although these cases seem to be examples of design errors and are considered to be design errors by at least some employees of the firm, a political system perspective provides a better explanation for these exceptions than an error perspective does.

Dynamic Organizations. The sourcing and scheduling information produced by computer systems is frequently incorrect. Although sourcing and scheduling exceptions could be classified as operation or design errors, a better explanation of the incorrect information is organizational changes. These computer systems were not in error when they were first installed. For sourcing information, as the company grew and facilities were both added and consolidated, the decision rules for assigning order components to plants also changed. Although the database of facilities and components was updated to match the new environment, a major restructuring of the database and programs was required to implement the current company decision rules.

Until that restructuring and reprogramming could be performed, plant assignment decisions made by the computer system using historical information and heuristics were incorrect. Similar circumstances applied to the scheduling computer system and the scheduling information it produced.

Organizational changes apply to people working in the process in a somewhat different way than they apply to computer systems because the people are performing exception handling. People will learn and develop routine procedures for detecting and handling exceptions as needed [Cohen 1991; Kling and Iacono 1984b]. For temporary or eliminated exceptions, execution of these procedures may persist long after there is any need to check for the presence of exceptions [Cohen and Bacdayan 1994]. For example, some special checking of orders from one sales region was being done even after the management in that sales region changed and the problems being checked for no longer occurred.

5.2 Political System Perspective

According to a political system perspective, differing and conflicting subunit goals can lead to exceptions that are difficult to eliminate. At our field site, relationships among customers, sales, and manufacturing generate examples of such exceptions.

Order processing in manufacturing regularly dealt with incorrect or incomplete administrative and technical information from sales. Although the sales organization, according to company policy, submitted only correct and complete orders to manufacturing, in practice, this policy was not enforced. Sales had incentives to submit "soft" orders to meet volume and revenue targets. These "soft" orders were for the correct product, but were missing details about product options (technical information). Additionally, manufacturing did not need all order information before starting production, e.g., the ship-to-address (administrative information) was not needed until the product was ready for shipment. In these cases, sales was supplying incomplete or inaccurate administrative or technical information, not because of mistakes (operation error), but because they wanted to get orders into the production pipeline.

Given sales incentives, manufacturing could not easily enforce a policy of accepting only complete orders. If such a policy were enforced, erroneous or inaccurate information would be supplied and then changed later. Manufacturing also had incentives to start production on incomplete orders if there was no order backlog. Unfortunately, the computer systems were designed to implement the official policy of complete orders (perhaps a design error), rather than the actual operating procedures of starting production on incomplete orders.

Changes to order information, another major type of exception, were generated by customers and salespeople. Customers requested different products, changed their desired delivery date, cancelled orders, etc. Although sales contracts stated that customers could not change orders within 30 days of promised delivery, this policy was unenforceable because customers could

always refuse to accept delivery of products. Eliminating exceptions caused by customers changing their orders required changing the behavior and expectations of customers. Although this could be done, organizations may prefer to respond to the requests of their customers by providing support for changing orders. Salespeople may submit changes to previously submitted "soft" orders, i.e., orders based on what they thought the customer wanted, but had not actually agreed to yet.

Priority orders represented a class of exceptions that involved a request to deviate from the standard process or procedure. Most of these requests were attempts to reduce the expected lead time, e.g., requests from customers or salespeople for priority treatment, continued processing without waiting for credit checks, and faster methods of shipment. If filling an order takes significant time, as is likely in a build-to-order process, inevitably there will be requests for special treatment for some orders. Although company policy was first-come, first-serve for orders, another policy, designed to limit sales requests for special order treatment, specified that a maximum of 10% of all orders could be priority orders.

The exceptions described above cannot be easily eliminated. In one sense, management tried to eliminate them by stating official policies (e.g., no changes to orders within 30 days of shipment) and then embedding these policies into computer systems. However, this served only to generate more exceptions and make exception handling more difficult and inefficient than necessary.

Using the actual operating policy rather than the official policy is not necessarily a feasible solution. Consider, for example, priority orders. The management of the firm was reasonable in publicly stating a first-come, first-serve order fulfillment policy and internally promoting equal treatment of customers. Salespeople also were reasonable in attempting to get priority treatment for their customers who wanted the product sooner than normal lead time. They were also reasonable in their attempts to meet revenue goals. Order processors recognized that it was in their interest and the firm's interest to be sure there were no problems with the orders of high-volume customers and critical accounts. Although the order processors who accepted priority orders worked in manufacturing, manufacturing also recognized the value of smooth and efficient production without priorities and changes after production started.

Management response to exceptions that fit within the political perspective was varied. For example, priority orders were capped at 10%, which seemed to be aggreeable to both sales and manufacturing. A code was added to orders to indicate that an order was a "soft" order so that sales could submit it, but manufacturing knew that the information was not necessarily accurate.

5.3 Total Quality Management (TQM) Perspective

A TQM approach of eliminating exceptions appeals to managers because exceptions require costly manual processing that results in reduced process performance, e.g., longer processing times, greater resource requirements,

and possibly lower output quality. The quality control literature, e.g., Fiegenbaum [1991] and Ishikawa [1985], argues that it is cheaper to fix the process than to fix continually the problems that occur. For computer systems that no longer match organizational decision rules, the resources allocated to exception handling represent the cost of out-of-date computer systems which, according to a TQM perspective, will be greater than the cost of updating the computer systems.

At our field site, configuration, sourcing, and scheduling information exceptions were generated by computer systems that did not match company policies. The firm was reworking the sourcing and scheduling systems and continued to maintain and update the configuration systems. The firm installed a version of the configuration expert system in the sales organization to help salespeople specify a technically correct product. The production system was being improved to eliminate gradually any remaining errors in transferring orders to it. Although our field site is updating its computer systems, there is a significant lag in developing and implementing these systems. This is not unusual; many firms have a large backlog of computer system development and maintenance requests [Swanson and Beath 1989].

These were technical, computer system maintenance solutions. Although they were relatively straightforward, they were neither simple nor inexpensive to implement. Keeping computer systems current with organizational policies and decision rules can be difficult. In the short term, it may be necessary to work around or handle computer system exceptions manually to accomplish work in a timely manner. These short-term solutions, however, seem to become long-term solutions. If the feedback loop to computer system maintainers is incomplete, maintainers may not get the information needed to initiate updates. Additionally, the availability of short-term work-arounds reduces the urgency of developing solutions.

Applying a TQM solution to political system exceptions is unlikely to solve the problems. For example, management at the field site has stated that (1) customers could not change orders, (2) sales could not submit incomplete orders, and (3) order fulfillment was a first-come, first-serve process. However, implementing these policies in computer systems (which some might believe would enforce the policies) has only made exception handling more costly rather than reducing the need for it. For example, it took more order processors and more processing time to make changes to orders when there were strict controls over changes. More experienced order processors were required to carry out the actual operating procedures for priority orders since priority lists were not documented.

In sum, a TQM solution of eliminating exceptions is likely to work best for exceptions generated by operation errors, design errors, or dynamic organizations. Some error exceptions can be eliminated by updating computer system capabilities. However, even this solution must consider that, for some complex and low-volume exceptions, people may be more efficient processors than computer systems. A TQM solution may worsen process performance when applied to exceptions generated by political systems because exceptions may

not be eliminated, but exception handling becomes more difficult, inefficient, and possibly error prone.

5.4 Human-Computer System Perspective

Although most of the process improvement efforts at our field site were consistent with a TQM approach, there were some examples of solutions that were consistent with the human-computer system perspective. For example, the configuration expert system correctly processed 98% of the orders presented to it, but it also provided support to the technicians for the remaining 2% of orders. The system "knew" when it could not generate a correct configuration given the components ordered and supplied technicians monitoring the system with information about why it could not complete the configuration. It then provided support for manual inputs to complete the configuration or to change the ordered components so the system could generate a configuration. The production computer system provided similar information about orders it did not accept, although it provided less support for changing orders to correct problems.

The firm was also providing more flexible reports and computer tools to assist order processors in monitoring the progress of orders. Although one goal of these efforts was to replace large nightly reports, this was a general move in the direction of an integrated human-computer process with order processors as process supervisors, rather than an automated process with order processors as exception handlers.

One example of a TQM approach at our field site resulted in a somewhat integrated human-computer system with cost-benefit aspects. The firm was attempting to reduce sourcing exceptions greatly by installing expert systems to catch and process the cases that the traditional system did not source correctly. Two expert systems were developed, one for midrange products and one for large products. The midrange expert system correctly processed approximately 98% of the orders presented to it, which met company policy of requiring at least 95% coverage before a system could go into production. This policy can be interpreted in light of a cost-benefit approach to mean that 100% coverage (i.e., no sourcing exceptions) may not be economically or technically feasible. In contrast to this policy, the large-products expert system was implemented with only 80% coverage. Large products were complex and low volume. The expert system handled the relatively routine orders, i.e., higher volume and lower complexity. People then focused on the most complex and low-volume (perhaps unique) orders that would be difficult to capture and maintain in an expert system. Although management was unhappy with 80% coverage, developers and order processors seem to have implicitly performed a cost-benefit analysis concluding that the 20% remaining exceptions were satisfactory.

The integrated human-computer system solution is best applied to those exceptions that organizations choose not to eliminate or cannot eliminate practically. One goal of this approach is to develop efficient methods for handling these cases, either by providing more efficient manual procedures or

by changing computer systems in a way that provides better support for the flexible processing that people can provide for complex cases.

In our view, each of these perspectives on exceptions provides some insight into the nature of exceptions in information processes. Focusing on any one is unlikely to address adequately the needs for improved performance of information processes. However, because some decisions are not programmable and some tasks, even if programmable, can be more cost effectively performed by people, a human-computer system perspective provides a useful, overall view for improving process performance.

6. CONCLUSIONS AND RECOMMENDATIONS

We have noted an inconsistency in the focus of research efforts between studies that clearly document the need for people in highly computerized routine processes and the many research streams devoted to learning to automate all human information processing. The current management zeal (around TQM) to eliminate all but random variation in organizational processes, especially routine processes, complements these computerization research streams.

We are not arguing against pushing the frontiers of computer capability, nor are we arguing against applying TQM approaches. There are clearly organizations and organizational processes whose performance could be significantly improved by applying TQM and process redesign techniques. However, to use continually advancing computing technologies in real organizations effectively, further research is needed in two general areas:

(1) the role of people in highly computerized processes,
(2) the design of computer-based systems that work effectively in organizational processes with multiple conflicting goals.

6.1 Directions for Future Research

During our field research, two problems related to the role of people in highly computerized processes became apparent. One was that people were expected to do whatever computer systems could not do; however, this role was poorly integrated with, and supported by, computer systems. This is an "irony of automation" [Bainbridge 1987]: designers seek to eliminate people from processes because they are unreliable and inefficient, yet they leave people to perform all the tasks the designer could not automate. The result is "an arbitrary collection of tasks, and little thought may have been given to providing support for them" [Bainbridge 1987, p. 272].

The other problem was a longer-term concern about maintaining and developing human expertise when computer systems processed most cases. People processed the most complex cases that computer systems could not process, but they did not have the opportunity to become experienced with simpler cases because computer systems processed these cases more efficiently. Additionally, employees needed typically to understand the decision rules incorporated into computer systems. Especially with expert systems,

problems exist with maintaining human expertise and becoming too dependent on expertise built into computer systems [Strong 1989; Sviokla 1990]. A partial solution may be to use expert systems for training and distributing expertise within the organization [Prietula and Simon 1989; Strong 1989].

The second general area for research, designing computer systems to function in the context of political processes with competing and conflicting goals, has been recognized as a critical research area for some time [Kling 1980; Markus 1983], but little progress has been made. In our view, a key component of such computer systems is flexibility, especially the ability to meet multiple and changing needs. Flexibility and the ability to change and adapt are key to user satisfaction with information systems [Bailey and Pearson 1983; Sterling 1974]. However, computer systems for operational-level processes generally have the effect of structuring and routinizing these processes [Markus 1984], even when advanced technologies such as expert systems are employed [Sviokla 1990]. At our field site, the flexibility needed to address conflicting goals adequately was provided by people working around the computerized process.

To support these two general research issues, some more specific research areas need to be further addressed, including:

—Methods for gathering and analyzing data on organizational processes and routines, i.e., how to take and analyze an organizational-level protocol.
—Computer support for exception detection and handling decisions (some initial progress is reported in [Strong 1992]).
—Ways for maintaining information systems efficiently and effectively. Much of the past research work on software maintenance has focused on the technical aspects of the software to be maintained [Bendifallah and Scacchi 1987]. Research is needed on managing maintenance [Swanson and Beath 1989] and on the relationship between system maintenance and the work process the system supports [Bendifallah and Scacchi 1987].
—Development of more flexible information systems.

Our research has only begun to touch on some of these research issues. Much research is still needed before we can design integrated human-computer processes that work well in dynamic organizations with multiple conflicting goals. However, using the observations from this study, organizations can take steps toward better computer-based information processes.

6.2 Managerial Recommendations

Organizations need to decide, based on a thorough understanding of the nature of exceptions in their processes, which exceptions should be eliminated and which exceptions represent the ability of an information process to handle a variety of requests. For problems to be eliminated, the root cause of the problems should be studied and then eliminated (i.e., take a TQM approach). For special cases, more efficient routines for detecting and handling them can be designed, e.g., develop computerized routines to handle them. The following five recommendations derived from our field study and

our perspectives on exceptions provide more specific suggestions for managing exceptions in routine organizational processes.

Recommendation 6.2.1. *Design mechanisms for evolving computer systems as the organization evolves.* The dynamic nature of organizational processes can result in computer systems gradually becoming mismatched to the organizational processes they supported. Users may find it easier to accommodate their work to the system rather than to negotiate and work with maintainers to fix the problems [Bendifallah and Scacchi 1987]. Additionally, MIS managers view user requests for maintenance as one of their key maintenance problems [Swanson and Beath 1989]. Better mechanisms for evolving computer systems as the organization evolves will rely on better communications and understanding among users, maintainers, and their management [Schneidewind 1987; Swanson and Beath 1989].

Recommendation 6.2.2. *Beware of radical process redesigns.* Attributing exceptions to process design problems leads managers to initiate process redesign projects, especially for processes that appear to be routine and well understood. However, these same routine processes are the ones that are critical for daily revenue generation and servicing of customers. Our field site experienced significant revenue declines when their order fulfillment process failed to move customer orders from sales into manufacturing after a major, yet relatively straightforward, reorganization. TQM recommends continuous improvement in small steps, which avoids some of the risk of process redesigns. However, performance improvements are likely to be smaller [Davenport and Short 1990], and process redesign experts argue that the large performance improvements from redesigns are worth the risk [Davenport 1993; Hammer 1990]. Although the reorganization at our field site may have been too large, a failure to appreciate the nature of exceptions explained by a political perspective on the relationship between sales and manufacturing in order fulfillment may have contributed to problems.

Recommendation 6.2.3. *Design more efficient exception-handling routines.* The three inefficiencies we observed in exception-handling routines, detecting exceptions by 100% inspection, unavailable information, and restrictive computer controls, can be addressed. First, good exception detection methods focus attention on exception instances, or at least on those cases that are likely to have problems. One function of computer systems is to focus attention on problems and possible causes of problems [Simon 1973; 1977], e.g., with exception reports. People can also be focusing mechanisms; e.g., salespeople indicated whether or not orders they submitted were "soft" orders. Second, information needed for exception handling could be made available in computer systems.

Third, restrictive controls should be evaluated to balance the need for adequate controls against support needs for exception handling. Controls in computer systems are generally good design practice, but they should match the actual controls used in organizations. More flexible controls in computer systems could provide a basis for resolving some exceptions best explained by

the political system perspective. For example, sales and manufacturing could negotiate a method, supported by computer systems, for easily tracking and changing incomplete or tentative orders, that does not overly penalize the performance of either group. If the computer system "knew" when each piece of information in the order was needed, it could provide controls so that sales could change information until the time manufacturing started taking actions based on that information. Although existing exception-handling routines can be made more efficient, a more global approach should be taken toward improving information process performance as is described in the next two recommendations.

Recommendation 6.2.4. *Design for people and computer systems, not just computer systems.* When computer systems are being developed for routine processes, designers tend to focus on designing the computerized routines—a natural focus for information systems analysts. A key aspect of designing integrated human-computer systems is to understand and evaluate the role of people in the process. The role of computer systems is generally clear; it is to process large volumes of information quickly and accurately. The role of people in high-volume transaction processes is less clear.

Designers need to design not only the computerized routines, but also routines to be performed by people, and the interaction between the two. Focusing on the interaction between computer systems and people is different from designing a user interface, which focuses on the computerized side. The interaction is important because it addresses the issues of adequate decision support, computer controls, and support for novices as well as experts.

Recommendation 6.2.5. *Design the entire process rather than focus on a functional area.* This recommendation further addresses exceptions attributable to the nature of political systems. At our field site, the computer systems provided better support for manufacturing's view of an ideal process rather than sales' view. However, since sales used the systems to meet their needs, manufacturing had to resolve more exceptions. Both groups would have better performance if the design supported the order fulfillment process rather than separately addressing the manufacturing and sales portions of the process.

Employing a cross-functional process view with a focus on customers rather than the usual functional "stovepipe" view of organizational work is a common TQM recommendation (e.g., Deming [1986]). However, this TQM recommendation still does not address conflicting goals. For example, two conflicting policies at our field site, (1) some orders had priorities and (2) order fulfillment was first-come, first-serve, were policies designed to address customer needs. Thus, we still do not have a good answer for designing a process in the presence of conflicting goals.

6.3 Conclusion

Although our in-depth, single-site field study achieved our research goals, data from a single site, necessarily, limits the generalizability of the results.

Our recommendations are most likely to apply to processes similar to the one we studied, i.e., operational-level, structured, computer-supported information processes. Additionally, the methods we used to collect our field data are traditional systems analysis methods that have limitations noted earlier that may lead to inadequate process understanding, e.g., cognitive limits of expert workers and process observers. Thus, our findings may be overly structured and rationalized.

Both of the general areas discussed for future research, the role of people in highly computerized processes and the design of computer-based systems that work effectively in organizational processes with multiple conflicting goals, are aspects of the design of organizational processes in conjunction with the design of computer-based systems. Further research in these areas is needed to provide the theoretical foundation for designing integrated human-computer systems that work effectively in real organizational processes.

ACKNOWLEDGMENTS

The authors thank Lester Diamond for coding work observation data, Lee Sproull for her many helpful suggestions throughout this study, and colleagues at Boston University for commenting on this article. We also thank the editor and several anonymous reviewers for their insightful and constructive comments.

REFERENCES

ANDERSON, J. R. 1980. *Cognitive Psychology and Its Implications*. W. H. Freeman, San Francisco, Calif.

BAILEY, J. E. AND PEARSON, S. W. 1983. Development of a tool for measuring and analyzing computer user satisfaction. *Manage. Sci. 29*, 5 (May), 530–545.

BAINBRIDGE, L. 1987. Ironies of automation. In *New Technology and Human Error*, J. Rasmussen, K. Duncan, and J. Leplat, Eds. John Wiley and Sons, New York, 271–283.

BENBASAT, I., GOLDSTEIN, D. K., AND MEAD, M. 1987. The case research strategy in studies of information systems. *MIS Q. 11*, 3 (Sept.), 369–386.

BENDIFALLAH, S. AND SCACCHI, W. 1987. Understanding software maintenance work. *IEEE Trans. Softw. Eng. SE-13*, 3 (Mar.), 311–323.

CASE, K. E. 1987. Quality control and assurance. In *Production Handbook*, J. A. White, Ed., 4th ed. John Wiley and Sons, New York.

COHEN, M. D. 1991. Individual learning and organizational routine: Emerging connections. *Org. Sci. 2*, 1 (Feb.), 135–139.

COHEN, M. D. AND BACDAYAN, P. 1994. Organizational routines are stored as procedural memory: Evidence from a laboratory study. *Org. Sci. 5*, 4 (Nov.), 554–568.

COHEN, R. M. AND MAY, J. H. 1992. An application-based agenda for encorporating OR into an AI design environment for facility design. *Eur. J. Oper. Res. 63*, 254–270.

COHEN, R. M. AND STRONG, D. M. 1991. A model for supporting database design. In *Proceedings of the 1st Workshop on Information Technologies and Systems*. Cambridge, Mass., 243–273.

DAVENPORT, T. H. 1993. *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press, Boston, Mass.

DAVENPORT, T. H. AND SHORT, J. E. 1990. The new industrial engineering: Information technology and business process redesign. *Sloan Manag. Rev. 31*, 4 (Summer), 11–27.

DEMING, W. 1986. *Out of Crisis*. Center for Advanced Engineering Study, Massachusetts Inst. of Technology, Cambridge, Mass.

ERICSSON, K. A. AND SIMON, H. A. 1984. *Protocol Analysis*. MIT Press, Cambridge, Mass.

FIEGENBAUM, A. V. 1991. *Total Quality Control*, 4th ed., Revised. McGraw-Hill, New York.

FRANZ, C. R. AND ROBEY, D. 1984. An investigation of user-led system design: Rational and political perspectives. *Commun. ACM 27*, 12 (Dec.), 1202–1217.

GALBRAITH, J. R. 1977. *Organization Design*. Addison-Wesley, Reading, Mass.

GALBRAITH, J. R. 1973. *The Design of Complex Organizations*. Addison-Wesley, Reading, Mass.

GASSER, L. 1986. The integration of computing and routine work. *ACM Trans. Office Inf. Syst. 4*, 3 (July), 205–225.

GOLDSTEIN, R. C. AND STOREY, V. C. 1991. The role of commonsense in database design. In *Proceedings of the 1st Workshop on Information Technologies and Systems*. Cambridge, Mass., 141–150.

HAMMER, M. 1990. Reengineering work: Don't automate, obliterate. *Harvard Bus. Rev. 68*, 4 (July–Aug.), 104–112.

ISHIKAWA, K. 1985. *What is Total Quality Control? The Japanese Way*. Prentice-Hall, Englewood Cliffs, N.J.

JURAN, J. M. 1989. *Juran on Leadership for Quality: An Executive Handbook*. The Free Press, New York.

KLING, R. 1980. Social analyses of computing: Theoretical perspectives in recent empirical research. *ACM Comput. Surv. 12*, 1 (Mar.), 61–110.

KLING, R. AND IACONO, S. 1984a. The control of information systems developments after implementation. *Commun. ACM 27*, 12 (Dec.), 1218–1226.

KLING, R. AND IACONO, S. 1984b. Computing as an occasion for social control. *J. Soc. Iss. 40*, 3, 77–96.

LENAT, D. B., GUHA, R. V., PITTMAN, K., PRATT, D., AND SHEPHERD, M. 1990. CYC: Toward programs with common sense. *Commun. ACM 33*, 8 (Aug.), 30–49.

MARCH, J. G. AND SIMON, H. A. 1958. *Organizations*. John Wiley and Sons, New York.

MARKUS, M. L. 1984. *Systems in Organizations: Bugs + Features*. Pitman Publishing, Marshfield, Mass.

MARKUS, M. L. 1983. Power, politics, and MIS implementation. *Commun. ACM 26*, 6 (June), 430–444.

MARKUS, M. L. AND ROBEY, D. 1988. Information technology and organizational change: Causal structure in theory and research. *Manag. Sci. 34*, 5 (May), 583–598.

NELSON, R. R. AND WINTER, S. G. 1982. *An Evolutionary Theory of Economic Change*. Harvard University Press, Cambridge, Mass.

NISBETT, R. E. AND WILSON, T. D. 1977. Telling more than we know: Verbal reports on mental processes. *Psychol. Rev. 84*, 3, 231–259.

PRIETULA, M. J. AND SIMON, H. A. 1989. The experts in your midst. *Harvard Bus. Rev. 67*, 1 (Jan.–Feb.), 120–124.

RASMUSSEN, J., DUNCAN, K., AND LEPLAT, J. (Eds.). 1987. *New Technology and Human Error*. John Wiley and Sons, New York.

SASSO, W. C., OLSON, J. R., AND MERTEN, A. G. 1987. The practice of office analysis: Objectives, obstacles, and opportunities. *IEEE Office Know. Eng. 1*, 1, 11–24.

SCHNEIDEWIND, N. F. 1987. The state of software maintenance. *IEEE Trans. Softw. Eng. SE-13*, 3 (Mar.), 303–310.

SIMON, H. A. 1981. *The Sciences of the Artificial*, 2nd ed. MIT Press, Cambridge, Mass.

SIMON, H. A. 1977. *The New Science of Management Decision*. Prentice-Hall, Englewood Cliffs, NJ.

SIMON, H. A. 1973. Applying information technology to organization design. *Pub. Admin. Rev.* (May/June), 268–278.

SIRBU, M., SCHOICHET, S., KUNIN, J. S., HAMMER, M., AND SUTHERLAND, J. 1984. OAM: An office analysis methodology. *Behav. Inf. Tech. 3*, 1, 25–39.

STERLING, T. D. 1974. Guidelines for humanizing computerized information systems: A report from Stanley House. *Commun. ACM 17*, 11 (Nov.), 609–613.

STINCHCOMBE, A. L. 1990. *Information and Organizations*. University of California Press, Berkeley, Calif.

STOREY, V. C. AND GOLDSTEIN, R. C. 1993. Knowledge-based approaches to database design. *MIS Q. 17*, 1 (Mar.), 25–46.

STRONG, D. M. 1992. Decision support for exception handling and quality control in office operations. *Dec. Supp. Syst. 8*, 3 (July), 217–227.

STRONG, D. M. 1989. Integration of expert systems, traditional systems, and people: A manufacturing example. In *Proceedings of the 3rd International Conference on Expert Systems and the Leading Edge in Production and Operations Management*. Univ. of South Carolina, Columbia, S.C., 57–69.

SUCHMAN, L. A. 1983. Office procedure as practical action: Models of work and system design. *ACM Trans. Office Inf. Syst. 1*, 4 (Oct.), 320–328.

SVIOKLA, J. J. 1990. An examination of the impact of expert systems on the firm: The case of XCON. *MIS Q. 14*, 2 (June), 126–140.

SWANSON, E. B. AND BEATH, C. M. 1989. *Maintaining Information Sysems in Organizations*. John Wiley and Sons, Chichester, England.

TODD, P. AND BENBASAT, I. 1987. Process tracing methods in decision support systems research: Exploring the black box. *MIS Q. 11*, 4 (Dec.), 492–512.

WHITTEN, J., BENTLEY, L. D., AND BARLOW, V. M. 1989. *Systems Analysis and Design Methods*, 2nd ed. Irwin, Homewood, Ill.

Received January 1993; accepted March 1994