Productivity in the Information Systems Department

An Honors Thesis (ID 499)

by

Elizabeth A. Rutherford

Thesis Director

Ball State University
Muncie, Indiana
May, 1982

Expected date of graduation (Spring 1982)
# TABLE OF CONTENTS

**INTRODUCTION** .................................................. 1

**DEFINING PRODUCTIVITY** ......................................... 4

A Historical View of Productivity
The 1980's Approach to Productivity
General Productivity Definition
Productivity Defined: Individual vs. Company
Productivity Defined: Efficiency and Effectiveness

**MEASURING PRODUCTIVITY** .......................................... 10

Why Measure Productivity
Productivity Measurement Model
Lines of Code
Other Measures

**IMPROVING PRODUCTIVITY** ......................................... 17

Selection, Training and Professional Development
Turnover
Management
Motivation
Programmer Productivity Aids
Using a Consultant
Summary

**SYNOPSIS** ............................................................. 34
Introduction

"Cullinane's ADS/Online cuts application development time by 90%.

Virtual Systems Software, Inc.'s CICS Interactive COBOL Symbolic Debugging System can "find and correct 98% of all CICS COBOL program coding errors in a single debugging session" and results in "more than a 1000% increase in programmer productivity."

ASI offers a computer-enhanced training system to help increase your people productivity.

The James Martin World Seminar will "put the pieces together" in a dynamic, information-packed, 5-day conference.

The sampling of recent advertisements from computer-related magazines and newspapers either amazes or amuses the reader. But it does indicate the current focus on productivity in the information systems (IS) field.

There are three reasons for this increased push for productivity. First, we, as a department, company, and nation, are bothered by the idea that we may be falling behind in some competitive race. In order to stay on top, we feel the need to buy new packages and push our people harder.

The second reason deals with the ever-increasing costs (especially the "people costs") of information systems services. It is only natural, then, to want to raise the productivity of the IS department, to "get more" for the money.
The third reason behind the current focus on productivity improvement is the backlog of application requests in almost every IS department. This problem is due to several factors: the increasing complexity of the systems being developed (note the changing focus from accounting and computing systems to decision-support and information-moving systems), the increasing quantity of system requests, the high turnover rate of IS personnel, the increasing amount of time spent on maintaining older systems, and the shortage of qualified personnel. These factors combine to form a seemingly infinite backlog of requests.

Much has been written on the topic of productivity. One can look at hardware performance, software efficiency, "people productivity," or any combination of the three. This thesis will be limited to discussing the various facets of "people productivity" in the IS department. We will discuss some of the factors, opinions, controversies, and examples that deal with programmer productivity.
DEFINING PRODUCTIVITY
A Historical View of Productivity

Before we try to define productivity today, let us look at the recent past. Steven Keider (19) provides just such a view in the March 30, 1981, issue of Computerworld. He states that in the 1960's, productivity meant substituting capital for labor, i.e., "How many payroll clerks can we replace with an IBM 1401?" The 1970's provided the opportunity to systematize the IS department. HIPO, structured programming, chief programmer teams and similar techniques and methodologies were introduced. Both of these decades saw "quantum leaps" in productivity, but Keider feels that these increases were chance by-products rather than a focus on productivity. He states that the 80's issue is a productivity issue, which means "the right person at the right time on the right job, but more important, doing the right thing and being measured accordingly."

What Mr. Keider fails to see, however, is that during the 60's and 70's the concentration was on productivity—on doing more work with fewer resources. The substitution and systematization that occurred were simply the methods used then to increase productivity.

The 1980's Approach to Productivity

What are we using in the 80's? A threefold approach to productivity can be seen today:

* prepackaged systems
* management style changes
* motivation
Software packages are getting a great deal of attention in the marketplace today. These packages are designed to make the programmer's job easier and/or quicker. (Perusing any issue of Computerworld will convince you of this fact.) However, buyers must remain aware of the fact that packages are not a panacea for all of the department's productivity problems.

The second precept gaining attention recently is that of "Americanized Japanese-style" management. Much has been written on how American businesses can improve their productivity by imitating the Japanese, whose management style places a strong emphasis on the worker.

The subject of motivation completes the list of issues for the 80's. In many magazines one find articles re-working Herzberg's, Maslow's and Skinner's theories, combined with surveys and hypotheses of today.

General Productivity Definition

Let us now address the question, "What is Productivity?" Several sources (8, 16, 19, 24, 26, 28, 29) give definitions, but the basic definition is an "output/input ratio," where outputs are goods and services produced, and inputs are resources used to produce output. (24) The government's strictly quantitative measure divides the rate of output by the rate of consumption of labor, capital, and materials. (28)

Problems arise in the IS area when one tries to specify what is meant by inputs and outputs. Inputs, or resources used, are the equipment and other materials costs, IS personnel costs and user personnel costs. The problem that presents itself here is one of dollar cost versus actual value. It cannot be said of any of the three types of
inputs that the dollar value expended on the equipment, materials, or personnel is equal to the value that he, she, or it contributes.

The outputs involved in this productivity ratio are even more difficult to define in an IS department. Shall we count completed screens, programs or systems? Rather, lines of code generated per man-day or man-month is a common benchmark, but unless properly administered, is generally useless.

Keider (19) suggests using direct corporate savings resulting from the implementation of a system as the output measures. This measurement would be very useful; however, finding the savings resulting from each system would be a difficult task. The entire concept of measuring output will be addressed in more detail later in the discussion.

**Productivity Defined: Individual vs Company**

Productivity can also be viewed from the individual versus the company standpoints. According to Samuel Connor (8) individual productivity is affected by four factors:

* the individual employee
* the manager
* the departmental setting
* the organizational setting

Each has both a direct and independent impact on an individual's productivity and also influences the other three factors.

The employee's own ability, determined by experience, training, and mental and physical capability, is a large factor in productivity. His own personal motivation is also a key element.

The manager affects his subordinates by planning well, defining specific goals and by minimizing changes and revisions in projects that are partially completed.
The departmental environment, including the presence of support systems and up-to-date reference materials, can greatly increase productivity.

The final factor, the organizational setting, describes the loyalty felt and the communication between the IS personnel and the remainder of the company. With the other three factors, organizational setting makes up the individual productivity.

We can also look at the productivity of the IS department within the entire enterprise. This focus implies the need to evaluate the total cost of handling a transaction or piece of information. (29) It sometimes happens that a systems team that develops a complex, state-of-the-art system is praised and commended, while no one notices the adverse effects on the "bottom line." (19) An emphasis needs to be placed on what the IS department does as a part of the organization, not as a single entity.

**Productivity Defined: Efficiency and Effectiveness**

Two other concepts that need to be discussed are efficiency and effectiveness. Efficiency refers to getting a desired result from a minimum expenditure of input or obtaining the maximum output from a specified amount of input. Effectiveness, on the other hand, is not concerned with input, but rather, refers to achieving defined results or objectives. (29) The following lists of measures for efficiency and effectiveness will serve to clarify these differences further:

**Efficiency Measures**

* On-time delivery
* Within budget completion
* Efficiency, in terms of CPU time, core used, and user response time
* Reliability of the production version
* Conformance to specifications (specs)
Effectiveness Measures

- Design that gets the job done (preferably the simplest possible system that accomplishes task or tasks)
- User satisfaction with end product
- Ease of modification
- Readability
- User training
- Documentation (16)

Productivity, sometimes equated with performance, encompasses both efficiency and effectiveness. For different projects, one may be more important than the other, but in all cases, a balance between the two must be found.

One final point concerning the definition of productivity. When a company decides to plan and implement a productivity program, it is essential that productivity be defined and understood by those who will be affected by the program. Too often, when asked to increase productivity, managers will respond, "My people are already working as hard as they can." We need to erase the negative associations with productivity and communicate the fact that management does not want them to work harder, per se, but rather, to work smarter.
MEASURING PRODUCTIVITY
Why Measure Productivity?

We have defined productivity as a ratio of outputs to inputs, from an individual and company-wide viewpoint, and as a combination of efficiency and effectiveness. Now we need to address the question, "Why measure productivity?"

Peter Lamasney (21) argues that we do not want to measure productivity in the information systems field. He feels that the field is more an art than a science, and that trying to define and measure certain accomplishments will only lead to more of whatever is being measured and less creativity in the work of IS personnel.

Mr. Lamasney's point holds true if the productivity program places an emphasis only on completing certain tasks. Certainly then, most people would try to accomplish more in terms of those tasks. But if the program has as its objective better management, the chances for success are greater.

Better management is one of the best justifications for being concerned with productivity. Simply by being involved with planning and implementing a productivity program, the manager will be made more aware of the individual functions of his department. This awareness, combined with the data collected, will aid in estimating time and money for future projects and also help in determining progress in the productivity of the department.

This rationale can be broken down into four basic reasons for attempting to measure productivity:
1) To provide a basis for comparison in evaluating new programming methodologies.
2) To provide a historical basis for estimating schedules and costs.
3) To provide management with relevant data on the performance of the IS function.
4) To develop a common terminology in the industry in order to increase the communication of new ideas and past experiences. (16)

In addition, measurements of some type are needed to aid in a reward system and to assure that the company is getting their money's worth. (26)

The overall objective of performance measurement, then, is to encourage better individual and team behavior and thereby improve the performance of the department as a whole, and to create a historical data base to be used in estimating time needed and costs for future projects. The achievement of this goal will do much to alleviate the "image problems" that the IS department has had by helping to generate better estimates that are based on historical data.

How does one go about measuring productivity? What are some of the inherent problems therein? What factors inhibit productivity in the IS department and what can be done? These are the questions which will be addressed in the remainder of this thesis.

Productivity Measurement Model

Measurement of productivity has been addressed by many articles (1, 15, 16, 20, 25, 26), but still no solutions have been found. Many of the articles entitled "Measuring Productivity" skirt the edges of the topic, but never give the lesson on "how-to." It is my conclusion, then, that no good method exists of measuring "white-collar" productivity. At best, one can follow the prescribed techniques and then combine these findings with the intuitive feeling of whether or not someone is doing
his job well. Do not expect, then, the "how-to" lesson, but rather let us gather some insight from the experts' thoughts on productivity.

Carl G. Thor, vice president of the American Productivity Center, urges the use of a four point model for productivity measurement and improvement. As the keynote speaker at the second annual Conference on EDP Performance Management, he discussed this model which provides a framework around which a productivity program can focus.

This four-step process involves awareness, measurement, appraisal, and improvement. (15) In the awareness step, a manager or department must recognize that productivity is a problem. Measurement of some set of criteria is then a prerequisite to solving the problem. Thor urges the use of Nominal Group Technique (see reference 15 for explanation) to arrive at this set of productivity measures. The important aspect of these measures, though, is that they are easily understood and clearly communicated to those who are going to be measured. Another suggestion (28) is to use job descriptions, written by both the employee and the manager, as an aid in developing evaluation criteria.

Once the criteria have been determined, measurement must take place. This includes observing performance and documenting that performance as compared to the criteria.

The appraisal step involves reviewing the problems and opportunities within the business entity being studied. The use of the measured criteria for the IS department is essential, but the goals and objectives of the company must also be considered.

Finally, the improvement step uses the knowledge of the problems and opportunities plus the resources available to make the desired changes.
Lines of Code

Thor's model is well and fine, but it does not show any of the problems involved in the measurement process. The major problem lies in the fact that the IS industry has no standard unit product. One cannot count systems completed, screens designed or "bugs" found and expect to use these measures as productivity guides. LOC (Lines of Code produced), which is used in many installations, is regarded by some as meaningless. Questions arise as to the "rules" of LOC: Does it include throwaway code? Code that was added or deleted because of specification changes? What about the differences due to the language used and the complexity of the system?

The controversy rages on. Capers Jones, assistant director of programming measurement at ITT's Programming Technology Center, feels that the use of LOC penalizes the use of high-level languages in a direct relationship to the level of the language. He cites this example: A program could be written in assembler with 10,000 lines of code in 22 months. This yields 455 LOC per month. The same program could be written in PL/I and have only 2500 lines of code. If it took 12 months to produce, the LOC per month would be 208. Finally, the same program in APL would have only 1000 lines of code and take only 9 months to develop. The LOC per month here is 111. (1)

On the other side of the coin, a thorough commentary exists in the August, 1980, EDP Performance Review. (16) Although many people see LOC as a productivity measure for an individual, this article argues that it should be considered a measure of the entire development function. As for the problems that arise concerning what to count and what not to count, the answer lies in making a decision and sticking
to it. The problems with complexity, language and the type and size of program are also addressed by the article, which then states that these may not be significant because of the averaging effect of the many different programs involved.

Thus, LOC is not a perfect measure of productivity by any means. But, "productivity is only productivity if it is measurable." (16) And currently, lines of code is the best available measure for the output of an IS group. If decisions are made concerning the counting rules, and if LOC is used to measure the productivity of an entire group as opposed to an individual, historical data could be collected and used effectively as a productivity measure.

Other Measures

One must remember that no new measure of productivity is going to be useful immediately. Some kind of historical data base must be gathered so that comparisons may be made. Table 1, taken from the September, 1980, EDP Performance Review shows eleven measures and where they can be found.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program (Module) ID &amp; Description</td>
<td>Manual</td>
</tr>
<tr>
<td>Milestone Dates</td>
<td>Project Management</td>
</tr>
<tr>
<td>Hours Charged by Development Activity</td>
<td>Manual</td>
</tr>
<tr>
<td>Number of Compiles</td>
<td>Job Accounting</td>
</tr>
<tr>
<td>Number of Test Runs</td>
<td>Job Accounting</td>
</tr>
<tr>
<td>Computer Usage</td>
<td>Job Accounting</td>
</tr>
<tr>
<td>Number of Test Cases</td>
<td>Manual/Job Accounting</td>
</tr>
<tr>
<td>Number of Errors Found</td>
<td>Manual</td>
</tr>
<tr>
<td>Lines of Code</td>
<td>Compiler Listing</td>
</tr>
<tr>
<td>Costs</td>
<td>Job Accounting/Manual</td>
</tr>
<tr>
<td>Quantity/Types of Problems Found</td>
<td>Problem Reporting</td>
</tr>
</tbody>
</table>
With this data, a number of other measures can be derived, such as:

- Cost per module
- LOC per module
- Errors per module
- Errors per LOC
- Cost/time by development activity
- Standards for module size, development time, etc.
- LOC per hour
- Problems found in production
- Problems per LOC

These quantitative measurements can be used to estimate the cost and time that will be needed for future projects, to assess the effectiveness of changes in methodology, and to provide quantitative data for both management reporting and exception reporting. (17)

Paul Meyer, in the October, 1980, issue of Data Management, suggests some other evaluation criteria. He states that professionals should be evaluated according to their ability to:

- Meet projected target dates,
- Learn and apply new technology,
- Gain user acceptance of work produced, and
- Effectively document systems and programs. (25)

In a nutshell, if these four criteria are being met, the professional is doing his job. However, note that this says nothing about whether he is working at 20 percent or 80 percent of his potential.

It should be obvious that measurement is essential to improving productivity—if only to know whether or not is has improved. It is also probably obvious that measurement of IS personnel is no easy task. It involves a dedication on the part of management to develop a set of performance criteria, to communicate the criteria to their subordinates, to gather the data, and then use it in a wise manner, keeping in mind that a productivity program's purpose is to improve the performance of the IS function as a whole.
IMPROVING PRODUCTIVITY
Productivity improvement is on many people's minds today, and although great strides were taken in the past decade, even more improvement can be made in the 1980's. Thinking back to the definition of productivity as a ratio of outputs to inputs, one recalls that improving productivity implies getting more output for the same or a lesser amount of input. But as you will see, several of the techniques for improving productivity involve adding more input. In this case, productivity means getting a proportionately higher amount of output.

Six different topics dealing with productivity improvement will be discussed here:

- selection, training and professional development
- turnover
- management
- motivation
- programmer productivity aids
- use of a consultant

It must be kept in mind, however, that nothing will bring instant results, but rather, productivity improvement requires a long-term dedication to whatever technique is chosen.

Selection, Training, and Professional Development

Although it may be difficult, a company must take measures to protect itself from hiring non-productive employees. (18) This careful initial selection that matches people against specific jobs sets up an environment for high productivity. Two steps to follow are checking references thoroughly and, more importantly, having an effective job interview. The interviewee needs to be told exactly what the job he
is interviewing for entails and what the skill requirements are. The interviewer needs to have a set of standard questions for each job type and specific questions that pertain to the system that the interviewee may be working with. (18)

Lack of knowledge may be a contributor to the productivity problem in an IS department. This problem is often corrected with the proper training and development programs. Training is the instruction a new employee receives when he first is employed. Part of this instruction is company oriented, such as corporate identity education and IS identity education (how the department fits in and where it is going).

The second part focuses on more technical knowledge: application education (what the major systems are and how they assist operating departments), training in a given language, in how to use certain packages or tools, such as a DBMS or an application generator, and system development courses.

Finally, some kind of instruction should be made (but often is not) in human relations and communications skills. (30) Each of these three different types of education is important in making the new employee feel comfortable and in creating a sense of "belonging" and loyalty to the company.

When IS employees begin to feel they are becoming obsolete, job satisfaction and self-motivation are eroded. They begin to look for other jobs and often this leads to decreased productivity. A professional development program should be part of a company's tactics in keeping its employees. The program must be goal oriented, linked to realistic and achievable career growth opportunities, integrated with the way the IS department develops and maintains systems, and tied to the business plan, goals and objectives of the organization as a whole. (30)
The goals of a development plan include the initial training of entry-level personnel, but extend beyond those first months into professional growth and career development. A professional development program attempts to provide some of the challenge that IS personnel need and to build "esprit de corps" in the organization, thus giving people a reason to stay. Other goals of the program include ensuring that employees are aware of management's expectations, providing the opportunity to work on challenging jobs, letting employees know the results of their efforts and communicating advancement opportunities in the organization.

Together, then, these three tactics, careful initial selection, a thorough training program and an on-going professional development plan, can assist in getting and keeping good employees. In addition, the knowledge level of employees who are in these programs is likely to be high and thereby, with a corresponding ability level, will contribute to increased productivity.

Turnover

"Cut turnover by 25% and you'll increase productivity far more than all the structured analysis, design and programming techniques ever will!" (30) Whether this claim is true or not, the fact remains that the IS industry has a high turnover rate, and that this turnover is one of the leading contributors to decreased productivity.

According to Kim Carlyle (5), the average length of employment is 18 months. Now consider this example: Say a programmer is hired for $25,000 and a recruiting agency was paid $6,250. After one year the programmer gets a raise of 10 percent. Fringe benefits are 25
percent annually. If this hypothetical programmer stayed the average 18
months, the total cost would be as follows:

<table>
<thead>
<tr>
<th></th>
<th>1st year</th>
<th>2nd year (6 mos. with 10% raise)</th>
<th>total salary paid</th>
<th>25% benefits</th>
<th>18 mos. salary and benefits paid</th>
<th>agency fee</th>
<th>18 mos. total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st year</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$25,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13,750</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25,750</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9,688</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48,438</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6,250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$54,688</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These cost figures do not even include factors for the low pro-
ductivity of a new employee or the productivity lost while another mem-
ber of the staff spent time with the new programmer. When these are
included with the above figures, one can see how expensive and non-
productive high turnover can be.

What is behind this high rate and what can be done about it?
Many opinions are offered as to the cause of high turnover (5, 8, 19, 30),
but most agree that contrary to what managers believe, money increases
are not a major factor in deciding to change jobs. More likely reasons
include poor interpersonal relationships with the manager, unclear job
assignments, no opportunity for growth (either personal or career) and
boredom. (8)

If one stops to think about these problems, then the solutions
are clear: more manager attempts at a relationship, clearer job assign-
ments, better training and professional development programs, career op-
portunities and challenging work assignments. The previous section on
training and professional development addresses some of these problems,
and the following sections on management and motivation address the
others.

Turnover, then, seems to be a contributing factor to productivity
losses, but there is no simple solution. One must look to other solutions
to the productivity problem and attempt to make one's employees want to remain.

Management

Under this heading of management, several sub-topics come under consideration. Should a position called Productivity Manager be created? Is the structure of the programming job all wrong? Should we imitate the Japanese? Is prototyping the answer? It is toward these subjects that the following discussion will be directed.

Five years ago, the position of productivity manager did not exist. Now many companies have established such a position. A wide variety of tasks is involved. These include establishing company-wide productivity goals and objectives, developing and maintaining a productivity plan, co-ordinating various departmental productivity objectives, and developing and maintaining productivity measurements. The person chosen to fill a position of this type also needs a wide variety of skills, ranging from systems management and computer technology to manufacturing and human resources development. (10)

I, however, feel that the creation of a small productivity group would better handle the tasks involved. Brought together from various parts of the company, these people would be better equipped, in terms of background knowledge, to develop and coordinate productivity goals.

There is a hypothesis that behind general programmer job dissatisfaction lurks the programming job that is constructed in such a way that it is incongruous to normal human intrinsic motivation. Two factors contribute to this incongruity.

First, there is the problem of responsibility. Some programmers have a very low opinion of the role of responsibility in their
jobs. (23) The job may be structured such that the programmer takes a system, does his respective task, and then hands it to another person, much like an assembly line. The programmer may never hear about the system again. This structure, it would seem, might contribute to the lack of personal responsibility felt for work done.

Author Philip Kraft (28) argues that techniques such as structured programming and team programming cause a loss of interest in the programming job. Managers who strive for simplification, he said, are pushing to standardize programming along industrial lines. The result is a programmer who is very good at one thing. This loss of skill and control by programmers leads to the "clericalization" of programming.

The answer to the structure problem lies in better communication between managers and programmers. Managers need to communicate more results back to the programmer, as well as making him responsible for changes that may come up. Communication on both sides is needed to address the simplification issue. Programmers should let their manager know what tools they need, yet should also be open to suggestions from the manager concerning the trial of new ideas for productivity improvement.

The Japanese-style management techniques have been getting a lot of attention recently. Why? Because Japan appears to be "beating" us at the productivity game.

The so-called Japanese-style management focuses on four things: creating a "family" spirit, a concern for the quality of a given product, teamwork and participatory decision making, and job rotation. The creation of the family spirit is accomplished through a genuine concern for
the employee as a total human being. The concern for quality is manifested in the quality circle concept—small groups of employees who meet to discuss and identify recurrent quality problems. Then, when one is found, the group seeks to correct the source of the quality problem.

Participatory decision making is a key element in the Japanese approach to running a company. Everyone who will be affected by a decision has the opportunity to be involved in the decision process. This kind of decision making process takes longer initially but leads to better acceptance and quicker implementation of the decision once made.

Job rotation can fight the boredom mentioned in the turnover section, and subsequently, keep more people from leaving. It also helps one learn about the company as a whole. One problem with job rotation in the IS department is that the programmer who spends six months in the marketing department may feel as if he were losing his state-of-the-art knowledge, which translates into decreased value in the job market. (3, 22)

On the other hand, Henry Nanjo, director of the city of San Francisco's DP department, claims that we cannot import Japan's techniques. The reason? Their culture is much more homogenous than ours. Here, he said, in a team management meeting, little is accomplished because of Americans' ingrained attitudes of individualism. (11)

If we look closely at Japan's basic techniques, we notice that several are re-writes of older motivation theories. A manager needs to pay attention to and focus on the individual by setting up some kind of quality circle to discuss productivity and by using participatory decision-making. Finally, job rotation needs to be accepted as a valid
concept and then needs to be put into action in order to increase productivity of the department.

Prototyping attempts to copy after other fields and develop a working model as quickly as possible and then revise it until the model suits the user's needs. According to an article in the September, 1981, issue of the EDP Analyzer, the benefits are less maintenance requirements and fewer design errors and oversights. Its general purpose is to test assumptions of user requirements, design architecture and even program logic. (4)

Since the user is supposedly involved from the start of the revising process, it seems that this would be a good way to reduce later errors. However, Dan Rosenburg (15) argues the traditional life cycle of a system versus the streamlined approach. Rosenburg, of Price, Waterhouse and Co., stated that traditionally, too little time was spent on the development phase of the cycle, resulting in large amounts of time being spent on maintaining and enhancing the system.

Personally, I can see where prototypes would have their use in testing out assumptions, but when a prototype shifts over into the production mode, an error has been made. The users have been denied the system's analysis time that has proven to be valuable in reducing time spent debugging and maintaining. (The more time spent in analysis and design, the less time spent testing and debugging.)

A good manager must be aware of all these factors and how they affect his department. He can survey the structure of the programming job and learn what would apply in his situation from the Japanese. He must also be a part of the decision concerning the use of prototyping and streamlined development.
One final bit of advice says that a manager should observe his best performers. It may be that some of their methodologies or habits that make them so productive would be transferable to others in order to make them more productive. (23)

Motivation

The most thorough study on the internal motivation of programmers was done by J. Daniel Couger and Robert A. Zawacki and published in the December, 1978, issue of Datamation. Using a measurement instrument that found the degree of satisfaction on five core job dimensions (skill variety, task identity, task significance, autonomy, and feedback from job), they compared the results of data processing professionals with a large database of information gathered by other researchers using the same instrument. Their four major findings are given below.

1) In examining the core job dimensions, the researchers found that programmers rated their jobs lower than other professionals rated their own jobs. In particular, skill variety, task significance and autonomy were common causes of low satisfaction.

2) In comparing the resultant "psychological states" (the state of the individual dependent on the degree of satisfaction with the core job dimensions), it was found that, as expected, programmers experienced a lower "experienced meaningfulness," "experienced responsibility," and "knowledge of results" from their work.

3) After defining growth need as "a strong need for personal accomplishment - for learning and developing beyond where one is now - for being stimulated and challenged," Couger and Zawacki found that the growth need among IS professionals was very high.
4) Social needs are lower for IS professionals than for any of the other base data personnel. Couger and Zawacki also note:

We are emulating the industrial engineers of the 1940's, who kept chipping away at each job on the auto assembly line until it was splintered into the ultimate levels of specialization. We seem to be concentrating just as fervently on fragmenting the jobs of analysis and programming. (23)

So what does all of this tell us about intrinsic motivation? We must look at each individual and determine what personal qualities he has that can be "triggered" to motivate him. In general, skill variety (perhaps provided by job rotation), task significance (a function of assigned responsibility and the feedback channels), autonomy, and personal growth (perhaps via a professional development program and career opportunities) are what programmers indicate would make their jobs more satisfying. The fourth finding of Couger and Zawacki contains a warning to managers to avoid forcing social situations and to be careful when putting programmers into teams. With some people, communication should be kept at a constructive minimum.

Daniel Yankelovitch, writing in Psychology Today, has an interesting viewpoint on motivation. He sees a "new breed" of employee as having come into being, thus requiring changes in the traditional motivation theories for the 80's. This new type of employee, he feels, is not motivated by money or success, and thus the employer cannot use these "perks" as threats or rewards to get results. Instead, the "new breed" highly values leisure, status and less depersonalization. (23)

This viewpoint is valid only to a point, in my opinion. Yes, status and leisure time are more important today than they used to be, but money is still a factor in motivation.
When the IS professions were first starting, money was a strong motivating factor behind the droves of people coming into the profession. Currently, many authorities believe that money is more of an attention-getter than an external motivator. (23) In other words, a high salary may draw people, but it will not contribute to their staying with the company. An additional point to consider is that even though the dollars may not be a motivator, a high salary and the luxury items that can be purchased with those dollars are status symbols—-and status is a motivator. Stated simply, money is not a cure-all, but rather one of the many tools for motivating employees.

What, then, can a manager do to motivate his employees? He needs to encourage a working atmosphere that values challenge and accomplishment. The work itself needs to motivate people. Opportunities for growth from magazines to seminars need to be emphasized. The manager needs to make an attempt to have some social interaction and to help the employee feel a part of both the department and the company. Finally, he needs to guard against too much job fragmentation by maintaining or even adding responsibilities.

**Programmer Productivity Aides**

A number of tools and techniques are available to aid in the programming and testing process. They include application generators, test data generators, precompilers, program analyzers and optimizers, documentation aids, librarians, report generators and translators. Nearly all of these tools are designed to reduce the time spent in coding and debugging programs, thereby improving productivity.

One problem with these techniques is that many reduce the level of knowledge needed to perform complex functions. Is this a move that
the IS industry wants to make? After all of the talk about professional development and growth, this author feels that improper use of some of these packages can eventually lead to lower productivity by reducing the overall intelligence level of our IS personnel. On the other hand, however, the software products can improve productivity if used to reduce the tedious and repetitive aspects of application development.

Much of the work being done in this area can fall under the heading application generators. This concept began in the early 1950's with the development of high level languages. Reasons given in the early Fortran specification document sum up the justifications for the new languages:

Production of programs automatically for IBM 704
Virtual elimination of coding and debugging
Reduction of application development time
Increased computer throughput
Feasibility to tackle new mathematical applications. (27)

These same reasons could, for the most part, be used today in justifying application generators, which basically anticipate the requirements of a variety of applications to the extent that those requirements are common to each application. In addition, an application generator provides programming language capabilities to adapt those parts of the application specific to the particular system. (12) These generators should only be used after the user of the generator completely understands the problem to be solved. There could develop a tendency for the programmer to jump into generating code before the problem is completely analyzed. This lack of analysis must be avoided.

A personal endorsement of COBOL application generators, particularly Phoenix Systems, Inc.'s System-80 COBOL generation system, was given in the May 21, 1981, issue of Computerworld. Rich Gemmuso,
supervisor of programming and financial systems for the U.S. Division of General Refractories, Inc., explained that using highly trained programmers to write and debug COBOL programs all day caused them to become bored, feel unchallenged and be less productive. He added that after adding the generation system, the shop no longer had those problems because they spent their time on the creative, analytical work that makes the job interesting. Addressing those who feel that a generator cannot do everything a programmer can do, Gemmuso said, "If a generator can give me 80% to 85% of the code I need in a quarter of the time it would take me to produce it, the time I've saved more than justifies the use of a generator." (7)

The problem I see with this school of thought is that the quality of the systems developed will deteriorate. If, in this system, only 80 percent of the code is generated, will the other 20 percent be generated? Even so, will the system be what the user required? It must be remembered that the user requirements are the final consideration of productivity.

Another advantage to application generators, as stated by Dr. George Schussel at the National Packaged Software Conference, is cost. The average package is $30,000 to $40,000, about 1/10 of what it would cost to develop it in-house. Additionally, maintenance is shared with the vendor. (2)

Cincom's Series 80 MANTIS is a new product designed to provide beginning-to-end on-line application development without any requirement for batch processing. This is supposedly an improvement over IBM's CICS Development Management System, which allows interactive screen design but requires table updates to be done in batch. The ad
in the April 26, 1981, issue of Computerworld (6) claims that MANTIS is a combination of a flexible, modern procedural language and the power of an interpretive system. The system includes:

* free form interactive screen design
* conversational file design and allocation
* logical data views
* conversational programming with MANTIS' 4th generation language
* on-line documentation
* computer aided instruction facilities
* master terminals security control
* menu driven operation
* independence to access any existing files or data bases
* on-line hard copy
* dynamic screen attribute modification
* full set of batch utilities
* high level Cincom DBMS and VSAM file support.

At the Clinton Corn Processing Company (a division of Standard Brands), David Beert has said that they use MANTIS as an aid in distributive processing. Engineers can now do some of their own simple requests. He said, "MANTIS has freed us from having to write CICS applications and it allows non-programmers to use the system." (29)

Another type of productivity aid is a testing and debugging aid. Since these functions can take 50 to 70 percent of a programmer's time, it makes sense to focus on these in trying to improve productivity.

XPEDITER is one such debugging tool from Application Development Systems, Inc. It was designed to increase programmer productivity by eliminating or expediting testing and debugging tasks. XPEDITER's Compile Processor (CP) extracts data from the compile output and puts it in more easily read forms. The Compile Processor (CP) also sets up a symbolic data set that contains attributes of each data element in the program, as well as the locations of each procedural label and source number. This allows the CP to reference data elements by source name and line number. CP
reduces test time by providing an automatic attribute update whenever a change in program code is made. The price for the package is $25,000. (32)

One other idea for improving productivity is to buy or set up libraries of re-usable code. R. G. Lanergan of Raytheon did just that and then went a step further. His group went on to develop a library support system that describes and indexes these modules so programmers can easily find out if what they need is already available. In five major applications consisting of over 250 programs, he has realized more than 60 percent re-usable code. This means a substantial savings in time and effort. (11)

A "shell" is one other kind of re-usable code exemplified by the Ball State Data Base Group. Here they set up a shell program of statements common to IMS programs. The programmer, then, fills in the logic specific to the application he is working on.

Just from this small sampling, one can see that many kinds of aids are available. And as "commonalities in programming solutions" are recognized, there will be a progression to even more automated tools. As previously stated, these tools can increase productivity and job satisfaction by large percentages if properly employed.

Using a Consultant

At times a fresh, unbiased viewpoint is needed in order to see productivity problems clearly. To provide this source of input, many consulting firms exist to offer help to a company. From executive seminars to long-term in-house consulting, one can buy an expert's advice.
For example, Dr. Martin Stankard, founder and director of the Productivity Development Group, is a consultant focusing on white-collar productivity and improvement. The objective of the Group is to "help clients obtain more results for less effort." Stankard has developed a three-stage approach involving diagnosis of the problems, drawing up a detailed improvement plan, and helping to implement that plan. Companies that follow his approach supposedly make one to three major productivity improvements. (A major improvement is either the replacement of labor intensive paper shuffling, reorganization, or getting back into "sync" with the marketplace.) (28)

Consultants using experience gained from other companies, can bring a new perspective to old problems. But again, consultants are not a panacea. Management is still responsible for implementing the needed changes.

Summary

We have surveyed the major methods of improving productivity, namely, selection, training and development, reducing turnover, improving management style, motivating the employees, using programmer productivity aids, and using a consultant. Each of these can be a part of a successful productivity improvement program, provided the needed dedication of management is present.
When we started this discussion of productivity, it was mentioned that different kinds could be studied. But when all is said and done, productivity is dependent on people. And "the real difficulty is that hardly anyone is willing to face up to, and put into words, the fundamental truth that what must be improved is people productivity." (13) This thesis has looked at several ways of improving the IS department's "people productivity."

First, productivity was defined from several points of view. An attempt was made to answer the question "Why study productivity?"

The second section of this paper looked at measuring productivity. The conclusion was that although no perfect method of measuring the IS department's productivity exists, some good approximations can be made. This data is most valuable if kept for a period of time and used for comparison purposes.

The third part of the discussion dealt with various ways of improving productivity. Thoughts on training and professional development, turnover, management, motivation, and programmer productivity aids, and consultants were given and evaluated.

One must not forget, though, in the rush of improving productivity, to remain focused on overall goals. In other words, one must not become too "tied up" in any particular methodology of productivity improvement. James A Walsh, corporate vice president and MIS director of CBS, Inc., (31) can be quoted as saying, "MIS productivity is not a single happening. It is the sum of all the things we do both good and
bad. Productivity is the end result of management practice." Featured in the May 18, 1981, issue of Information Systems News, the CBS installation has implemented many of the techniques discussed in this thesis. But the point that Mr. Walsh makes is that the sum of the changes and improvements must reflect favorably upon the profits of the business to be considered as productivity improvements. Again, a manager cannot become "overly enthusiastic" about a single method of improvement. Rather, he needs to keep the total picture in mind.

Computer futurist James Martin has predicted that by 1990 computers will require 93 times as much code as is being generated today, resulting in a need for 27.9 million programmers. And although this prediction is rather comical, it does serve to illustrate the need for a focus on productivity. This thesis is a mere tip of the iceberg in terms of information available. But reading alone will not solve the problem of IS productivity. Management must take what they read and apply the techniques in their own situation. Again, only via the dedication of all levels of management to a productivity improvement program that uses any or all of the methodologies described herein, will that program succeed in improving the productivity of the IS department and the company as a whole.
REFERENCES


10. "Demand for Productivity Czar," Infosystems, April, 1982, pg. 64.


