Software Engineering Institute

Concepts on Measuring the Benefits of Software Process Improvements

James A. Rozum

June 1993

TECHNICAL REPORT
CMU/SEI-93-TR-009
ESC-93-TR-186

Software Process Measurement Program

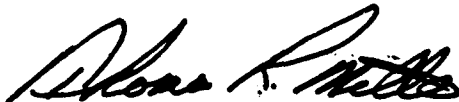**Carnegie Mellon**

This technical report was prepared for the

SEI Joint Program Office
ESC/ENS
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official
DoD position. It is published in the interest of scientific and technical
information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

Table of Contents

List of Figures

Preface

This technical report is intended to provide organizations with some concepts, measures, and methods that can be used to determine the benefits they realize from their software process improvement efforts. The report provides a set of recommendations for organizations that want to implement measures to determine their software process improvement benefits. The intent of the recommendations is to give an organization a starting point to determine the benefits of its software process improvement effort. Organizations that have a method for measuring the benefits of their improvement efforts can use the concepts to expand their measurements. After reading this report, an organization would decide on its intended approach and then continue planning its implementation of the measures and processes to implement them.

Additionally, some concepts in this report can be used to determine the benefits from using new technology; i.e., an organization does not have to have a comprehensive software process improvement effort to be able to use the concepts in this report.

The SEI receives daily inquiries for measurements that demonstrate the savings to be made by moving to higher levels of software process maturity. Other than a limited set of anecdotal articles and presentations, no such data set exists. This report proposes some concepts to begin structuring such a measurement program. If your organization has experience in using these or similar concepts for measuring the performance of your software process improvement effort and would like to share that experience, please contact:

Julia Allen
Manager, Industry Operations
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

(412) 268-6942 (voice)
(412) 268-5758 (FAX)
jha@sei.cmu.edu (Internet)

Acknowledgments

Concepts on Measuring the Benefits of a Software Process Improvement Effort

Abstract: The software community initially became aware of process improvement from the works of Deming, Juran, and Crosby. The current awareness and activity regarding software process improvement was sparked by the Software Engineering Institute (SEI) with the release of its original software process maturity model. Following the advice of the SEI, many software organizations initiated software process improvement efforts to improve the quality of their products by improving the processes that produce those products. The question that many managers are continually asking today, though, is: How much has my organization benefited from the changes we have made? Unfortunately, many organizations did not include a method of measuring those benefits in their improvement activities. This report describes some concepts that organizations can tailor and build upon to develop a method for determining the benefits they have received from their software process improvement activities.

1. Introduction

> "Software process is the set of activities, methods, and practices that are
> used in the production and evolution of software." [HUMPHREY]

A process can be as big and broad as the entire software development process, or as small and as focused as a process for design inspections. Regardless of its size, every process is an opportunity for improvement. Software process improvement can be any action taken to make a software process better than that which exists.

1.1 Benefits of Process Improvement

Early process improvement activity is typically targeted at stabilizing the current process or reducing waste. Waste is the single most important cause of dissatisfaction, low productivity, and excessive cost. The amount of waste in any given organization is larger than most believe or will admit. Experience has shown that waste amounts to between 20% and 50% of an organization's annual net income, with the average around 40% [CONWAY]. In other words, for every $1000 profit earned, an additional $400 may not be realized. Process improvement is intended to reduce and eliminate that waste and allow an organization to capture that $400. In addition to capturing that $400, organizations also find that once most of that $400 is captured, doors begin to open to even greater rewards and potentially larger returns, e.g., increased productivity or an expanding customer base from customers gravitating towards the higher quality products. These rewards and returns are considered the benefits of process improvement. And, although Conway's data are not directed at software specific organizations, from speaking with others in the software field, it is believed that Conway's numbers are low for software intensive organizations. This document is

intended to help organizations determine how to measure and quantify the benefits of software process improvement.

It is not clear from the literature where the greatest rewards lie for those heading down the path of software process improvement. Two organizations, Hughes Aircraft Company's Software Engineering Division and Raytheon Equipment Division's Software Systems Laboratory have both claimed tremendous success and returns from their improvement activities, yet neither has claimed that their organization is at the point of optimizing their organization's software processes [WILLIS], [SNYDER], [DION92a], and [DION92b].

As an organization strives to continuously improve its process, it reaps several rewards, as reported by Willis and Dion. The first set of rewards from stabilizing an ad hoc and undefined process is the decreased cost of producing better products. Often, this is evidenced in increased employee morale and pride in the organization. The organization is also rewarded when the results of its superior quality products and services become recognized and it experiences increased customer satisfaction and increased demand for its software products and services.

1.2 Immature Versus Mature Organizations

In process improvement, we sometimes speak of immature and mature organizations. A mature organization is one that has a foundation for improvement in place and is now continually looking for opportunities to improve its process.

An immature organization has processes that are usually undefined and ad hoc. These processes are not easily repeated and often unpredictable. Often, the immature organization is reactive to problems, often putting itself into a crisis or chaotic situation. The quality of the products produced by immature organizations is usually poor and never predictable; the cost and schedule of projects are usually overrun. These poor predictive capabilities are usually the result of a process that is not well defined and, hence, changing and often improvised. New technology is often looked upon by the organization as a silver bullet; but the organization is usually not able to use the technology effectively and, often, the new technology only makes matters worse. Successes achieved by an immature organization are usually the result of individual talent and heroic efforts [CURTIS] [HUMPHREY].

In contrast, mature organizations have defined and documented processes. Because the processes are repeatable and predictable, the organization is able to use those processes to plan efforts and monitor projects based on the plan and processes. Quality, cost, and schedule are predictable—within known tolerance levels—because the organization is using not only the processes it has documented, but is also using knowledge from past projects to estimate the current projects and to improve the organization's software process and products. The organization uses new technology only when a need arises, not when a technology arises. Often that need is to improve a specific process or aspect of a process. Success is predicted and expected. Failures are few and usually the result of outside influences. The mature organization that is continually improving is an organization that

produces exceptional quality products versus quality products that are the exception [CURTIS] [HUMPHREY].

Figure 1 illustrates how an organization might benefit by reducing the variance of its estimates [CURTIS]. In Figure 1, an organization that does not have a foundation for process improvement typically is at the initial level and has processes that are chaotic and unpredictable. Such an organization cannot adequately predict the outcome of its efforts and hence cannot predict the time or cost to produce its products. Therefore, the probability density function for its efforts has a large standard deviation. Additionally, the target value for the organization is often much less than the actual outcome.
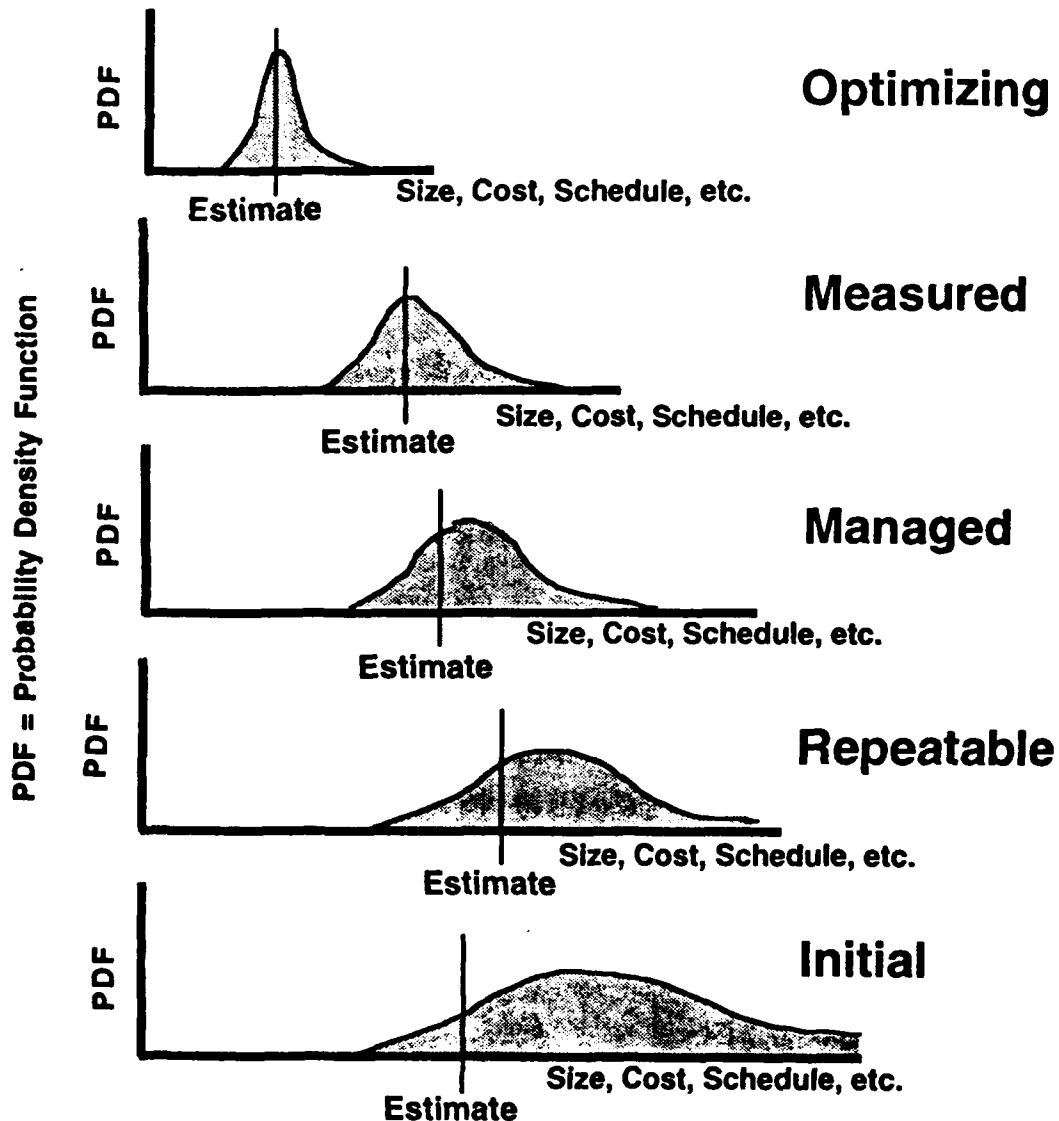
Figure 1 - Theoretical Ability to Predict Process Outcomes of Organizations

In Figure 1, as an organization establishes the foundation for continuous process improvement, the standard deviation decreases and the probability of accurately predicting the time and cost goes up accordingly. In other words, the probability increases that the target value is within a reasonable tolerance of the actual outcome. In addition, as an organization better understands its processes and is better able to predict the outcome of those processes, the organization begins to see opportunities to reduce the time and cost of its processes, i.e., improve its processes.

1.3 Measuring the Benefits of Process Improvement

To date, many organizations have implemented numerous software process improvements. The common questions being asked of those organizations are: What quantifying measures can be used to determine the progress of software process improvement efforts, and what effect have those efforts had on the organization? In short, quantitatively, how has an organization changed as a result of its software process improvement efforts? The purpose of this report is to identify some concepts that an organization can expand upon and use to establish its own measures for change resulting from its software process improvement effort.

Every organization that launches a software process improvement initiative must be able to articulate the qualitative and quantitative value added by this initiative. Middle and upper management require these data to determine if process improvement can or will contribute to meeting the business objectives of the organization, such as return on investment, decreased time to market, building a new product with fewer staff resources, or reduced cost of rework.

Commencing and sustaining a successful software process improvement effort requires a multiyear commitment at all levels of the organization; it is often labor intensive and difficult to institutionalize. The major difficulty for organizations at the initial maturity level of the CMM is defining measures and creating mechanisms to collect and analyze the data. It is these measures and mechanisms that will reveal the results of process improvement actions.

A word of caution to the reader is appropriate here. If your organization does not have methods in place to measure its improvement efforts, start small! All measurements and methodologies should be evaluated to determine their appropriateness to your organization and its overall software process improvement effort and tested prior to implementing them throughout your organization.

1.4 Scope and Overview

This report is intended to introduce some concepts that an organization might expand upon and use to show the benefits of its software process improvement activities. To help organizations that do not have measures in place for determining how they have changed as a result of their software process improvement efforts, this report includes some suggestions

on types of measures to start measuring the change. Also included are some suggested concepts that an organization can use to expand its measures and methods after the organization is comfortable with its basic measurements. Additionally, the concepts discussed can be used to measure the benefits received from using new technology; i.e., an organization does not have to have a comprehensive software process improvement effort to be able to use the concepts in this report.

No two organizations are alike; this implies that each software process improvement program will also be different. These differences may be large or small, but, because there are differences, each organization must decide how best to measure the benefits it receives from its software process improvement efforts. This technical report, therefore, discusses a range of possibilities and keeps its methods independent of models or methods for software process improvement, as well as specific models or methods for software measurement. Likewise, this technical report is not a guidebook that gives a step-by-step method on how every organization should measure the benefits or return-on-investment from its software process improvement effort.

Though organizations should use a conceptual model to guide their improvement efforts, software process improvement is the important point, not which model the organization uses as the guide. Organizations should pick, and probably tailor, a robust model that makes sense for them. All organizations are different and all improvement efforts are different. An organization must choose the measures that are best for it and tailor and define the measures accordingly.

In preparing this report, the author has drawn principles from other disciplines, e.g., operations research, business administration, and quantitative methods programs, as well as from specific, nonsoftware process improvement methods and the measuring of benefits regarding specific measurement models. The author has taken these principles, methods, measures, etc., and adapted them to illustrate how they can be used by a software process improvement effort.

1.5 Audience

The intended audience for this report is broad: organizations that either have a software process improvement effort underway or would like to start one. Those organizations that have an effort, but do not have a method for determining how changes to their software processes have improved the organization, should find the concepts in this report most useful in helping to choose a measurement method to implement. Organizations that do have a method for measuring the benefits of their improvement efforts should still find some thought-provoking concepts that they may wish to use to elaborate their existing measures. Organizations just starting a software process improvement effort can benefit by considering how to incorporate measurement into their efforts near the beginning of their software process improvement programs.

2. Determining the Cost and Savings of Software Process Improvement

Measuring the benefits of software process improvement is itself a process [ROZUM]. The process for determining the cost and savings of a software process improvement effort includes:

- Translating the organization's goals and objectives into key attributes of the products that the organization delivers to customers.

- Considering the software process improvement activities that support those attributes as assets of the organization. These activities should be connected to the organization's strategy for meeting its goals and objectives. Organizations must realize that not all assets provide a direct return on investment to the organization, but instead, some assets exist to enhance the value of other assets.

- Establishing a cost baseline of the software process.

- Determining the cost of the improvement activities undertaken.

- Determining an amount saved as a result of the software process improvement activities.

- Comparing and tracking the cost of the improvement activities versus the savings that result from the changes.

A software process improvement (SPI) benefit index might have as the numerator the difference in the cost of the software process from the previous process to the process that results from the improvement activities. This difference in the processes is the amount saved. The denominator would be the cost of the previous process, i.e., the process we are comparing the new process against. The resulting SPI benefit index would be:

$$\text{SPI benefit index} = \frac{\text{cost (old)} - \text{cost (new)}}{\text{cost (old)}} = \frac{\$ \text{ saved}}{\$ \text{ cost}} \quad \text{(eq. 1)}$$

The amount saved is measured by quantifying the difference in the cost to produce software before and after the improvements. These differences in costs, or savings, are usually the result of reduced rework, less effort to produce an item, increased productivity, reduced maintenance costs, etc. The performance of the process is measured and then those measures are normalized using a software size measure. Using a software size measure to normalize process performance, an organization can compare the performance of its software processes used to develop and/or maintain various software products.

Costs of the improvement activities are discussed as either nonrecurring costs or recurring costs. Nonrecurring costs are defined as those costs expended by the process improvement effort to establish a process or purchase an asset (for example, the cost of training personnel, hiring consultants for specific tasks, implementing changes, etc.) Recurring costs

are defined as the costs of the activities that have been implemented to monitor products, prevent errors, or continually guide or manage the effort.

Although the costs, for the most part, can be objectively measured, the dollar value of the amount saved is sometimes a mix of objective and subjective measures. Items such as reduced rework, productivity, effort, etc., can be objectively measured and, often, a distinct dollar value can be attached to changes. However, some measures of improvement, such as increased reliability or customer satisfaction, are sometimes difficult measures to quantify in terms of a dollar value. Because some measures are difficult to quantify, an organization should implement measures of its software process improvement effort in an evolutionary manner; that is, an organization should start small by quantifying common aspects of the process and concentrating initially on measures that are less difficult to define, collect, and use consistently.

The organization should only consider the savings and costs of activities that are a result of its software process improvement effort. Some changes may be made to the organization's software process that are necessary for the organization to compete in a certain area. An example might be the changes an organization makes to receive certification under the ISO 9000 series of international standards. In such cases, the organization should consciously decide whether or not to include those changes as part of its improvement efforts.

2.1 Determining Software Process Improvement Costs

Before an organization can measure the return on its investment from improving its process, it must be able to measure the cost of the investment, considering both nonrecurring costs and recurring costs.

2.1.1 Nonrecurring Costs

Nonrecurring costs are those expended to initiate the improvement effort or an improvement activity, to purchase or develop an asset for the effort (e.g., a tool), or to establish a process. Nonrecurring costs are usually one-time events. Examples included as nonrecurring costs are:

- Consultant fees.
- Initial training fees.
- Costs for developing or modifying standards or procedures.
- Initial planning costs, i.e., the costs for planning the process improvement effort.
- Development costs, i.e., the costs incurred to research and/or develop a new process or document an existing process so that it can be improved.
- Pilot-testing costs, i.e., the costs associated with testing new methods or techniques prior to implementing them in the process throughout the organization.
- Implementation costs, i.e., the costs required to change a process.

- Labor cost for doing an assessment of the organization's process.
- Development of a measurement database.

2.1.2 Recurring Costs

Recurring costs include money and resources expended toward activities that manage or support the ongoing software process improvement effort. Included in the recurring costs are:

- Overhead costs, i.e., the cost for the SPI effort expended by those who manage and control the ongoing effort (for example, the costs associated with staffing and maintaining a process group whose primary responsibility is facilitation of the SPI effort).
- Error-prevention costs, i.e., the cost incurred to analyze a process to determine where errors are being inserted and how to change the process to inhibit them.
- Process-monitoring costs, i.e., costs expended to measure the current process, analyze the data, and report on the analysis results.
- Error-detection costs, i.e., costs expended for processes to detect errors (for example, reviews, inspections, audits, etc.)

Examples of recurring costs include:

- Software engineering process group (SEPG) staff.
- Staff who trace their existence to an improvement recommendation, e.g., those involved with error tracking or causal analysis.
- Maintenance of a software measurement database. (This should not include the cost for developing the database. The costs for developing the database would be a nonrecurring cost.)
- External advisory board costs.
- Program reviews by corporate management.
- Design or code inspections.
- Audits of the software process.
- Continual training, e.g., training of new employees in the organization's process.

These are just a few examples, and, arguably, a cost identified as nonrecurring may be classified as recurring and vice versa. Initially, it is not as important where the cost is included, only that the organization include the cost in its calculations and consistently apply the categories over time. An organization should identify all costs relevant to its process improvement effort and determine how to capture those costs regularly.

2.2 Measuring Savings

After determining the cost incurred from the SPI effort, the organization determines the amount of money saved. Some of the less elaborate concepts used to determine the amount of money saved can be done by quantifying the dollar value of items such as:

- Increased productivity.
- Early error detection and correction.
- Overall reduction of errors.
- Improved trends in maintenance and warranty work.
- Eliminating processes or process steps.

Each is discussed in the following sections.

2.2.1 Increased Productivity

To measure the savings incurred from an increase in productivity, the organization must first be able to measure its productivity. Productivity is usually defined as the amount of product produced per units of resource used to produce that product. For example, a software organization might measure productivity monthly by lines of code per staff month. However, this simple equation is often exploited, easily misused, and rarely fully understood. To avoid misuse, rigid exit criteria should be maintained on when a product is completed, the counting rules should be consistently applied, and what is included and excluded in the resources used to produce the product should be well defined.

It can be misleading to compare the productivity observed of different products in an organization's product line, even though the same measurements are made. For example, it is not appropriate to compare the software productivity observed for a missile guidance system versus a payroll system because of the differences in complexity and criticality.

To avoid inappropriate comparisons, some organizations and cost estimation models have developed productivity indexes. These productivity indexes are simply equations that take into consideration the differences of the products and weight those differences as a way of normalizing them. The result is a unit-less index versus a number with defined units. For software development organizations, Putnam has developed such an index [PUTNAM90] [PUTNAM92]. Another method adopted by many software organizations is to normalize their software using function points. The function point methodology is a method maintained by the International Function Point Users Group (IFPUG) and can be used to normalize attributes of software projects [IFPUG90].

The key to determining a normalization method is picking a measure that is independent of factors such as technology, style, methodology, and the organization. These are the factors that will be affected by changes in the process. If the measure is dependent on these factors, then the changes may be difficult to determine because the data are influenced by

the measurement method; any correlation or trends in the data may be the result of the measurement methodology, not the changes in the process.

Figure 2 illustrates an example using a productivity index to determine the improvements in an organization's productivity. To determine the productivity benefit achieved by the process improvement effort, the organization would determine the dollars saved by increasing one increment on the productivity index. Each data point in Figure 2 would be the productivity index observed for a completed project. A straight line is then fitted to the data. There are a number of possible fitting techniques; the only constraint is to be consistent in whichever method is used.
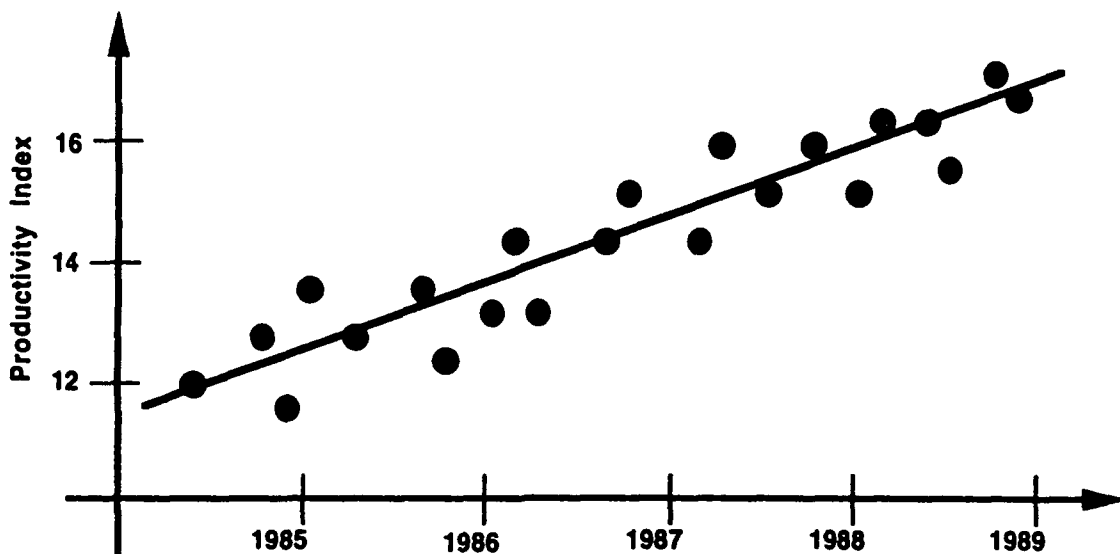
Figure 2 - Measuring Productivity Changes

Another improvement that is difficult to measure, but that sometimes shows up as an increase in productivity, is the benefits from maturing to a repeatable process. For more on this, see [HUMPHREY].

2.2.2 Early Error Detection and Correction

Figure 3 illustrates how the relative cost to correct a software requirements error increases as that error is perpetuated throughout the life cycle and detected in later life-cycle phases [BOEHM] [CURTIS]. One of the goals of the SPI effort should be to reduce the number of errors made and to detect and remove errors early in the process. Early detection and correction of errors is a major cost savings or avoidance of cost achieved by a SPI effort. To measure this savings, an organization must collect data on when in the process each error is detected and when in the process the error was made. This implies that error detection

processes such as reviews, testing, inspections, etc. need to be part of the process. Needed along with this data is the actual cost to correct an error found during each phase.
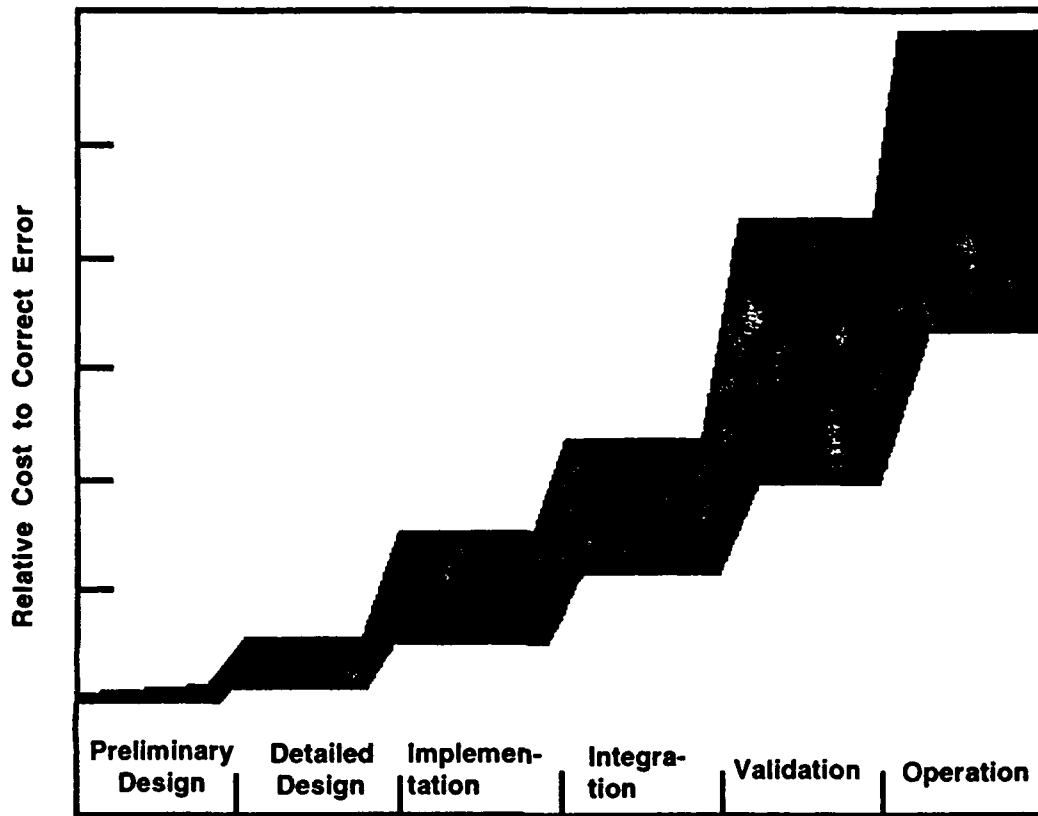
Figure 3 - Error Correction Costs by Phase

With the cost data on correcting errors, an organization can construct an indicator similar to Figure 3 for its software development process. For each phase in Figure 3, the organization would determine a mean cost for correcting errors detected in that phase. Using an indicator similar to Figure 4 for different time periods (e.g., consecutive years), in conjunction with the mean cost data, the organization can then estimate its savings from detecting errors earlier in the life cycle by determining the change in percentages of the errors found from one time period to the next. The indicator in Figure 4 can also be used to focus future improvement activities.
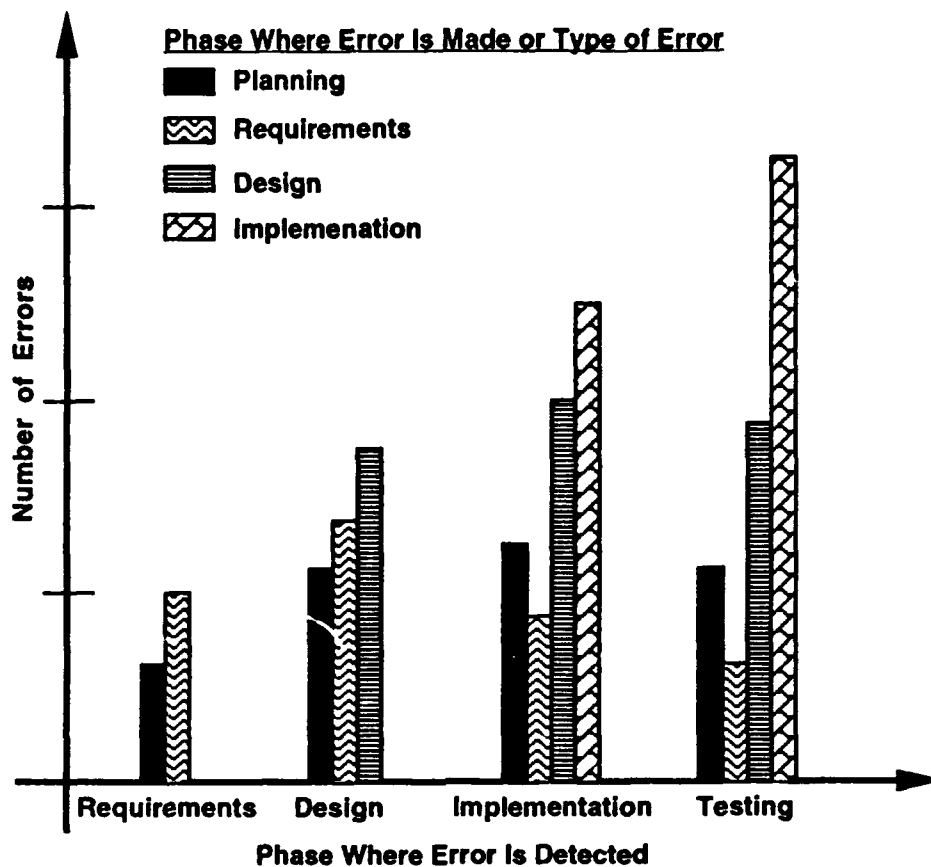
Figure 4 - Phase Where Errors Are Detected Versus Where They Are Made

Example 1 illustrates how to apply measurement to determine a benefit from process improvement in terms of making fewer errors and discovering those that are made sooner in the life cycle.

> **Example 1:** Consider the data below for the hypothetical corporation Acme Software. The data are the number of errors made during implementation from one of Acme Software's projects. After 1988, Acme implemented some changes to the implementation phase of its software process. Acme has previously calculated that, on this project, it costs two hundred dollars to fix an error both made and detected during implementation and that it costs five hundred dollars to fix an error made during implementation, but corrected and detected during the testing phase. To simplify this example, these costs remained the same for the years in question.

Acme Software data on Implementation errors of a project:

Phase Found	1988 Implementation Errors		1990 Implementation Errors	
	Number	Percent(%)	Number	Percent (%)
Implementation	40	40	54	60
Testing	60	60	36	40
Total	100		90	

From the data, we see that fewer implementation errors are being detected during testing in 1990 than in 1988 (20% overall, i.e., 60% versus 40%). This implies that if changes were not made to the process, an additional 18 implementation errors might have been expected to be found during the testing phase (20% of 90). Those 18 errors found one phase earlier saved $5,400. That is, if the improvements were not made, the cost of finding implementation errors in 1990, using the same detection percentages as 1988, would have cost $34,200 versus $28,800. This equates to a savings of 16% but does not take into account the additional amount saved from making 10% fewer implementation errors overall (90 errors in 1990 as compared to the 100 made in 1988).

2.2.3 Overall Reduction of Errors

Data on the total errors detected for a project, when normalized with size data, can be used by an organization to determine how well the organization's SPI efforts are improving the overall quality of its products (i.e., the reduction in the number of errors made during development). Using that information, the organization can determine the amount it is saving in rework costs. To determine this savings, an indicator similar to Figure 5 can be used.

To measure the benefit in dollars saved by reducing the number of errors, an organization uses data on the average number of staff-hours to repair an error along with information on the total number of errors and the average cost of a staff-hour. To determine the savings, the average number of staff-hours to repair an error is multiplied by the number of errors made during the time period in question. This result is then multiplied by the average cost of a staff-hour for the time period. The information is then compared to other time periods to determine the savings from the improvement activities. Data on the average cost to repair an error can be drawn from information used to generate the indicators in Figures 3 and 4.
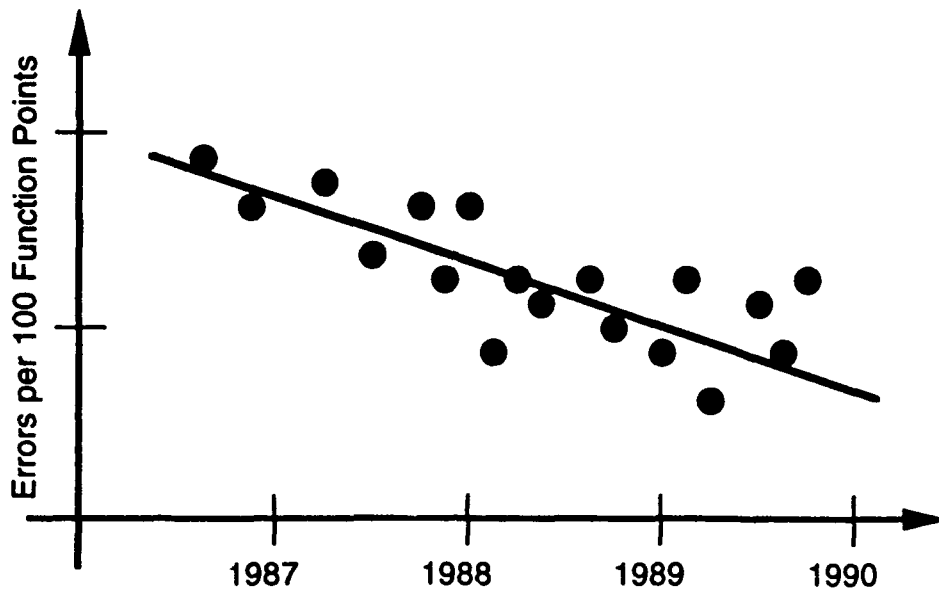
Figure 5 - Sample Errors Detected Per Unit of Size[1]

Example 2 illustrates how an organization can measure the benefits it receives from an overall reduction of errors.

> **Example 2:** Consider the data below for the hypothetical corporation Acme Software. Acme Software has implemented some improvements as part of its software process improvement initiative on four projects referred to as simply, A, B, C, and D. Assume the projects each have one major release per year that includes enhancements as well as the correction of errors. The four projects use principally the same software process and all have improvement initiatives applied equally. Acme has calculated its average cost per staff-hour for these types of projects as $50. Also, to simplify this problem, assume that each project completes an equal amount of work each year in terms of the amount and complexity of the functionality changes in each yearly release. (If this were not true, we would have to normalize the data to the size of the software.)

---

[1] In reality, the curves in Figures 5 and 6 are logarithmic decay functions that approach zero, but never actually reach it. For simplicity of the example, however, the figures are fitted and shown as simple x-y lines.

Acme Software error data for four projects:

Year	Project A	Project B	Project C	Project D
Total Errors Found				
1985	135	N/A[2]	N/A[3]	54
1986	121	34	N/A[3]	47
1987	67	64	78	41
1988	N/A[1]	61	62	N/A[4]
Average Staff-hours Expended / Error				
1985	250	N/A[2]	N/A[3]	233
1986	233	268	N/A[3]	171
1987	197	210	222	168
1988	N/A[1]	171	154	N/A[4]

Notes:
1) Project A ended 8/31/87
2) Project B started 7/1/86
3) Project C started 1/1/87
4) Project D ended 12/31/87

It costs Project A 250 staff-hours to correct an error in 1985. At an average cost of $50 per staff-hour and 250 staff-hours per error, Acme paid $1,687,500 to correct errors on Project A in 1985. Project A had similar costs of $1,409,650 in 1986 and a projected yearly cost of $994,850 in 1987. Project A has a calculated savings of $277,850 from 1985 to 1986 and a projected savings of $414,800 (i.e., based on the projected costs for a complete year) from 1986 to 1987; a projected total of $692,650 for the period. For all of the projects, the calculated savings are as follows:

Project Savings ($)

	A	B	C	D
1985-86	277,850	—	—	227,250
1986-87	414,800*	239,200*	—	57,450
1987-88	—	150,450	388,450	—
Total	692,650*	389,650*	388,450	284,700

* based on projected costs for a complete year

Combined, Acme Software has a projected savings of $1,755,450 from these four projects. This is just the savings calculated for reducing errors and the cost to correct an error.

By normalizing the project data with a software size measure and using an indicator similar to Figure 5, an organization can show the trend for projects that have implemented improvements to reduce the overall number of errors. To extrapolate the savings to the entire organization, the organization would calculate the average savings per unit of software size and then, using information on the size of other projects in the organization, determine a projected savings from implementing the improvements throughout the organization. Obviously, the organization can also further breakdown the error information to track savings and trends of various levels of errors, e.g., priority one errors versus priority two.

2.2.4 Improved Trends in Maintenance and Warranty Work

Another savings to take into account is the reduction in maintenance and/or warranty work that the organization performs on its products. Figure 6 illustrates how this data can be presented.
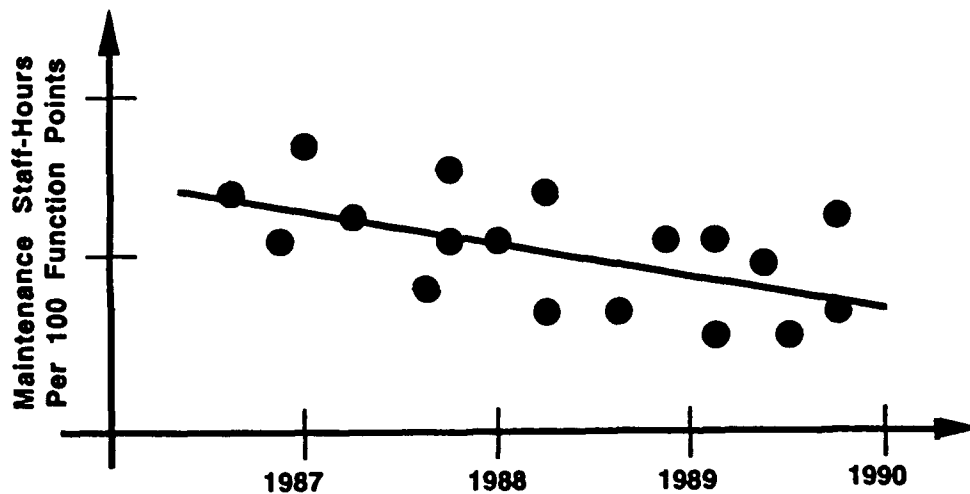
Figure 6 - Sample Maintenance Effort Trend

To determine the savings in corrective maintenance costs, an organization would determine the average number of staff-hours used per month for a given year (i.e., total staff-hours expended in a year divided by twelve) for maintaining each of its software systems. This number, the average monthly staff-hours, is then multiplied by the average cost of a staff-hour. The result is the average cost per month to maintain the software of a given system. This number can then be compared to other time periods to determine the savings from improvement activities.

Example 3 illustrates how an organization can measure the benefits received from its improvement effort by determining the reduced cost of maintaining its software.

> **Example 3:** Again, consider the four projects from Acme Software in Example 2, but with the staff-hour data below to maintain those four projects. In this example, however, we only look at the savings received from a decreasing level of effort

needed to complete an equal amount of work, ignoring the earlier data on errors. Assume these are the same four projects, that the same assumptions apply as in Example 2, and that Acme is still paying an average of $50 per staff-hour. Also for this example, we will assume that a project completes an equal amount of work each year. That is, a project is adding approximately the same number of enhancements each year and the total enhancements are approximately equal in size each year. (If this were not true, we would have to normalize the number of staff-hours each year to the size of the software.)

Acme Software staff-hour data for four maintenance projects

	Project A	Project B	Project C	Project D
Staff-hours Expended (x 100)				
1985	1,542	N/A[2]	N/A[3]	467
1986	1,341	987	N/A[3]	395
1987	809	882	838	310
1988	N/A[1]	694	702	N/A[4]

Notes: 1) Project A ended 8/31/87
2) Project B started 7/1/86
3) Project C started 1/1/87
4) Project D ended 12/31/87

To determine the savings, we calculate the changes in the number of staff-hours expended from year to year. Since all of the projects do not cover a complete year, the first step in this example is to calculate the average monthly expenditure of staff-hours for each project. Project A expended an average of 12,850 staff-hours per month in 1985 (154,200 / 12), 11,175 in 1986, and 10,112 in 1987. The year-to-year changes are then 1,675 fewer staff-hours per month in 1986 than in 1985 and 1,063 per month fewer in 1987 as compared to 1986. At $50 per staff-hour, Project A saves $83,750 per month (on average) in 1986 as compared to 1985, and an additional $53,150 per month in 1987 compared to 1986. This is a total savings of $1,005,000 for 1986 (12 * $83,750) and an additional projected savings of $637,800 for 1987. Project A would have a total projected savings of $1,642,800 for the time period in question. (Note that Project A had a real savings of $425,200 for 1987, though, because it ended in August of that year; in other words, Project A had a total real savings of $1,430,200.) For all of the projects, the yearly savings are:

Project savings ($)

	A	B	C	D
1985-86	1,005,000	—	—	360,000
1986-87	637,800*	799,800*	—	263,400
1987-88	—	940,200	679,800	—
Total	1,642,800*	1,740,000*	679,800	623,400

* based on projected costs for a complete year

Combined, Acme Software has a projected savings of $4,686,000 from these four projects because of the decreasing level of effort needed to complete an equal amount of work.

By normalizing the project data with a software size measure and using an indicator similar to Figure 6, the organization can show the trend for projects that have implemented improvements to reduce the level of effort needed to complete an equal amount of work. To extrapolate the savings to the entire organization, the organization would calculate the average savings per unit of software size and then, using information on the size of other projects in the organization, determine a projected savings from implementing the improvements throughout the organization.

Figure 6 could just as easily show the dollar value on the y-axis; however, a dollar in 1987 does not have the same value as a dollar in 1990, and true improvement trends might be disguised. For example, we may actually be improving by requiring less maintenance or by bettering our maintenance practice (depending on what is targeted for improvement), thereby requiring fewer staff-hours. But, if the cost of a staff-hour also rises, the end cost might be the same. How does an organization measure its savings then? Savings could be measured by determining the change in the required number of staff-hours and multiplying that change by the average cost of a staff-hour for the year within which the benefit is being measured. For example, in Figure 6, a straight line would be fitted to the data and then the organization would determine the change in the number of software maintenance staff-hours expended per 100 function points from 1987 to 1990. That number multiplied by the organization's average cost of a staff-hour in 1990 would be used as the amount saved over that time period.

A similar indicator can be used to track the number of service calls or other attributes of warranty work to determine the trend in the amount of warranty work an organization performs.

2.2.5 Eliminating Processes or Process Steps

As an organization matures, its improvement efforts will eliminate unneeded steps in its software process by either consolidating processes or replacing ineffective or inefficient processes. To determine savings from such changes in its software process, the

organization determines and compares the cost to perform the old process and the cost to perform the new process. To be able to make this comparison, both the old and new processes are normalized to a measure of software size. Sometimes, the change will result in a new process that is more expensive to perform, e.g., requiring more staff-hours to complete the same amount of work. However, an overall larger savings may be realized through increased productivity or less rework on the entire project. For example, an organization might add or replace an ineffective inspection process with a more expensive process that requires more people and/or more sophisticated detection tools. The goal, though, is for the new process to be more effective at finding errors, causing the amount of rework to drop, the machine maintenance and down time to decrease, and/or productivity to increase.

3. Recommendations for Starting to Measure Software Process Improvement Benefits

This chapter contains recommendations that an organization might consider when starting to implement measures for determining the benefits received from its process improvement efforts. These recommendations, when followed, are intended to establish a baseline from which future improvements can be measured. This chapter also discusses some recommendations for items to implement after a baseline is established. Towards the end of the chapter are recommendations for expanding the initial set of measures that can help an organization to improve its measurement of, and insight into, its improvement effort. The critical components to quantify the benefits, though, are the actions needed to establish a baseline for the current process and implement measures of the organization's performance. In short, an organization needs to initiate the regular collection of the following items:

- Staff-hours
- Errors
- Size data

Most organizations collect this data to some degree, even if it is at a rudimentary level. They need to build on that foundation, ensuring that the following basics are in place before expanding to other measures.

To successfully measure the benefits for the organization, all projects should report data using similar frameworks. This consistency is important for determining how the organization is changing at an aggregate level. A standardized work breakdown structure (WBS) for the organization is beneficial and can be used as a source for collecting effort and cost data. See [GOETHERT] for additional information on consistently reporting staff-hours, see [FLORAC] for reporting error data, and see [PARK] and [IFPUG92] for size data.

3.1 Collecting Staff-Hour and Error Data

Within the organization, each project would usually report staff-hours and errors monthly. The size data should be reported every quarter at a minimum. All three data items should be reported for every software release.[2]

Staff-hour data are collected for all employees of the organization's staff, as well as for subcontractor staff, for each software system being developed or maintained by the organization. The organization's staff-hour data should be collected and maintained in a

---

[2] In this instance, the term "release" refers to each time a software update is sent to customers or a baseline is established for a new version of the software.

management data reporting system. Subcontractor staff-hour data could also be stored within the same system.

Error data are collected on all products and processes of the organization. At a minimum, the organization should be able to determine where the error was found (product) and what process discovered the error. It would also be helpful to determine what process caused the error or initially contributed to its existence. Similar to the staff-hour data, the error data should also be kept in a management data reporting system.

3.2 Collecting Software Size Data

A measure of software size is required to normalize other measures. The size measure should be independent of technology, development methods and style, the organization, standards, etc., because the improvement effort will change these attributes and how the organization deals with them. For example, the organization may consider changing its design method or coding style, or implement tools to improve an aspect of the process. If the size measure also depends on these attributes, changes will be disguised and, when other measures are normalized, benefits will be either overstated or understated; in either case, the true benefit from the change will not be visible. The method for measuring size should also be well defined so that the variance in the results is minimal when the method is applied by different individuals. Because of these requirements, the most common software size measure, lines of code, might not be a viable measure to normalize project data to determine organizational benefits. Lines of code could be used, though, until the organization adopts a more viable method.

A viable size measure should be independent of the attributes mentioned above and also take into consideration the complexity of a system. The size data should be captured during the entire software development life cycle. Estimates are acceptable, but the data attributes of the estimates should be verifiable at the end of the development life cycle. Unfortunately, to date, such a size measure does not seem to exist, but measures such as function points [IFPUG90], [IFPUG92], feature points [JONES], and variations of Halstead's software measures [HALSTEAD] seem to be close. The organization should determine an interim method, possibly lines of code, and investigate a longer term solution.

Like the staff-hour and error data, the organization should also have a database for the software size data. In addition to actual data from the final system, the database should include estimates, decisions, and assumptions used to generate the estimates.

3.3 Procedures for Collecting Data

The organization should establish policies and procedures for the collection and reporting of its staff-hour, error, and software size data to ensure that data are consistently collected and

reported. These policies and procedures would provide the definitions of the data items to be collected and also identify the processes for collecting and reporting the data.

Regarding the specific policies, the size policy should direct projects to estimate and maintain current data on the size of its software systems. The error counting policy should include a definition of when an error is closed and a description of the various categories for the errors. When reporting staff-hour data, it is particularly important to define what types of project personnel are included in the counts. (For example, are support staff and management, as well as project functions such as design, testing, configuration management, or quality assurance included in the counts?) Each policy should include the process for collecting the data and maintaining a database as well as definitions of the key database fields.

The organization would typically establish a process for auditing itself against its policies. The audits should pick projects at random to verify . · ɟ validate that projects are conforming to the policies. As part of the auditing proce: ndividuals performing the audit may actually collect project data (e.g., software size da..), and use this information to determine discrepancies in data reported and variability in the counting process used by the projects. By verifying the data in this way, organizations can also use the audits to identify opportunities to improve the data collection process as well as to determine where additional training or improvement in training courses may be needed. The organization should also periodically audit its subcontractors for compliance to its policies. The audit findings should be reported to upper management, but reports must not identify individuals or projects.

3.4 Summary of Recommendations for Organizations Just Starting

An organization starting a measurement program would compile the following indicators:

- Staff-hours per unit of software size.
- Errors per unit of software size.
- Amount (size) of software released the past quarter.

Each of these indicators should be displayed as an aggregate of the organization's projects. That is, individual project data would not be identified. There are a number of ways the project data can be summarized, including:

- Combine all projects for the organization.
- Combine all projects within major divisions of the organization.
- Combine projects by platform, e.g., U2200.
- Combine projects by system type, e.g., business applications or missile guidance systems.

The errors per unit of software size could be displayed on an indicator similar to Figure 5. The staff-hours per unit of software size could be displayed using an indicator similar to Figure 6 for the organization's staff-hours and another, separate indicator for its

subcontractor staff-hours. The indicator to show the amount of software released would be similar to Figure 7.
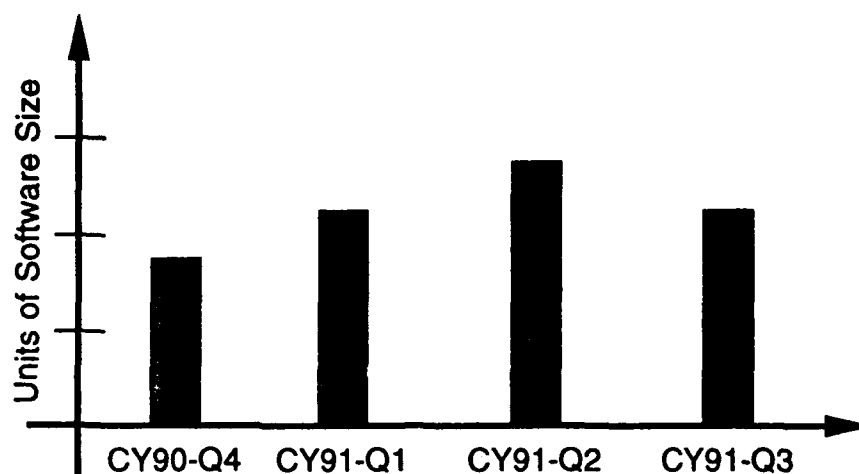
Figure 7 - Sample Amount of Software Released per Quarter

Initially, these indicators can be used to measure improvements at the project and then at the organizational level. For example, the indicators could be used to measure changes associated with the insertion of a single technology or improvement on a project. Then, the organization can use the project data and a size normalization factor to estimate the costs and savings if the entire organization were to insert the technology. The measures and indicators would then be used to track the actual performance of the improvement throughout the organization.

3.5 Recommendations for Expanding the Initial Set of Measures

Because a number of organizations already have the foundation in place for most of the above recommendations, implementation of the processes to collect the data consistently and regularly can be done quickly and a baseline can be established shortly after collection starts. Beyond the initial set, an organization can consider many other recommendations for additional measures of its improvement effort.

Figure 8 represents the layers needed to progress towards applying measurement to more complex issues. In Figure 8, an organization would begin basic data collection; that is, the organization would collect the three basic measures described in the earlier recommendations (staff-hour data, error data, and software size data) to begin measuring the benefits of software process improvement. The organization would then progress through the layers of the pyramid until it reaches the top and is able to quantify in dollars its return on investment (ROI).
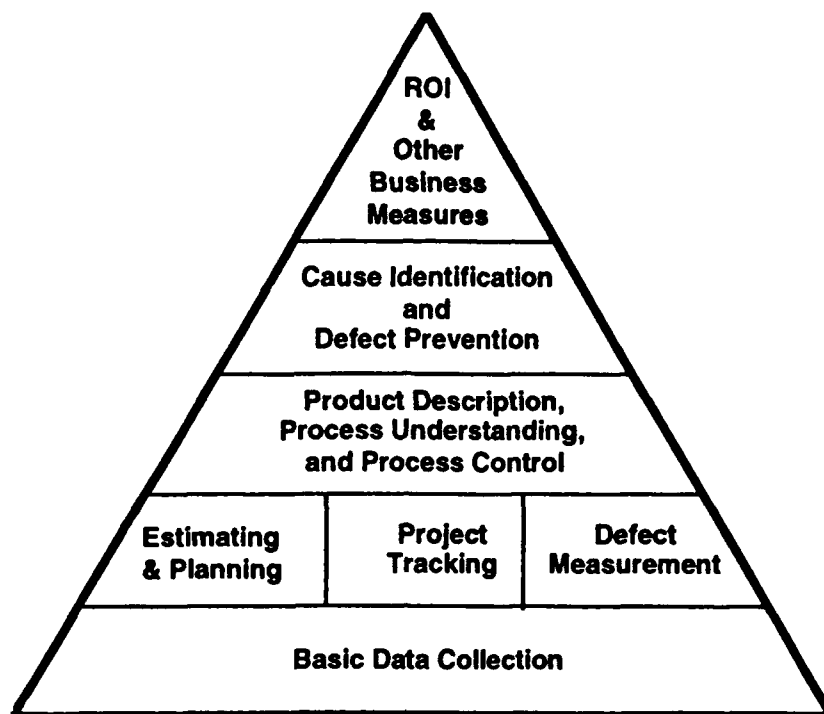
Figure 8 - Measurement Capability Pyramid

The first step to obtaining the capability of measuring ROI is to begin data collection of the initial data set and establish a baseline. Then, continuing up the pyramid, the organization would continually strive to improve the data and reporting methods for the initial data set through its estimation and planning, project tracking, and defect measurement processes. Particularly, the organization should concentrate on methods to validate the data being collected and reported. Having these processes stabilized and repeatable, the next steps would be to use an understanding of the process to:

- Determine the cost of the improvement effort in dollars.

- Determine dollar values for the changes in the indicators using the initial set.

Various methods for doing each of these steps are discussed in Chapter 4. While determining how to measure the cost and dollars saved, an organization would also choose a method for using the cost and savings data to formulate an index for determining the overall benefit to the organization. This index, then, is used later as a basis to determine the ROI of the improvement effort.

After the organization implements a process and methods for determining the cost of the improvement effort and dollar values for changes, the organization can consider expanding the initial set with other methods discussed in Chapter 2 and then more complex measures as discussed in Chapter 5. These measures that lead to expansion of the initial set are typically used to prevent defects and identify causes and effects.

Before continuing on toward upper levels of the pyramid in Figure 8, the organization should validate its data collection processes for each measure and method implemented and measure the variance of those processes. When the variances are at an acceptable level, the organization can continue implementing other measures as it deems necessary, continuing until it is able to determine its ROI or other business measures of its performance accurately.

4. Methods for Determining the Overall Benefit

Measuring the benefits of an organization's process improvement efforts is a question that some operations research or management science methods try to address. The term operations research or management science is commonly defined as the scientific approach to decision making, which seeks to determine how best to design and operate a system, usually under conditions requiring the allocation of scarce resources. We learn from operations research that to maximize an attribute of a system, we often operate some subsystems at less than optimum levels. This is a key principle for those who use data from measuring the benefits of improvement efforts. It implies that, although some improvements might appear to be less than optimal for a certain process, the improvements help to optimize the overall process of the organization [WINSTON].

The overall benefits received by the organization due to the SPI effort can be determined using a SPI benefit index. One example of a SPI benefit index is to use the number of dollars saved and the costs incurred to implement and maintain the SPI effort. The dollars saved is the number of dollars saved as a result of a change (or changes) identified and made by the SPI effort. (The savings would be calculated using one or more of the methods described in earlier chapters.) Likewise, the costs are the dollar values incurred to make the changes and to sustain the overall SPI effort. These costs are in addition to the cost of performing the existing process.

The dollar value of new business or additional income realized due to the SPI effort could also be added to the equation. However, it is often difficult to determine if the additional revenue is due to the improvements or some other feature, such as marketing, new products that were introduced, new versions of products released with new or improved features, increased exposure, etc. Unless the additional income can be objectively determined and directly tied to changes made or recommended by the SPI effort, it is probably better to track the data separately.

This chapter discusses two methods for determining the overall benefit obtained from the SPI effort: (1) a ratio of savings to cost (SPI Benefit Index Method), and (2) calculating the benefit of each change separately and totaling the results (Summation of Benefits Method). In either case, careful accounting of savings and costs should be made to ensure that items are not counted more than once.

4.1 SPI Benefit Index Method

As described earlier, cost is a function of the nonrecurring costs and the recurring costs, where the nonrecurring costs and recurring costs are simply summation functions of the costs within each category. The SPI cost equation that is used as the denominator of the SPI benefit index in eq. 1 (Chapter 2) is then:

$$\text{SPI Cost} = \Sigma(\text{nonrecurring costs}) + \Sigma(\text{recurring costs}) \qquad (\text{eq. 2})$$

Also described earlier are the various savings that can be measured. Assuming that the calculated savings are independent (or that those savings have not been counted twice), the equation for the numerator of the SPI benefit index in eq. 1 becomes:

SPI Savings = Σ[savings from(finding errors earlier + making fewer errors
+ making the process more efficient and/or effective + decreasing
warranty cost and/or maintenance + increasing productivity)]   (eq. 3)

The savings equation (eq. 3) and cost equation (eq. 2) combined then leads to eq. 4 (a form of eq. 1):

SPI Benefit Index = SPI Savings / SPI Cost   (eq. 4)

Note that in eq. 3, not all of the potential savings need to be included in the calculation. That is, the organization only includes savings it has calculated or can calculate in equation 3. The organization needs to decide what it will include in such a calculation and how it will determine those independent savings. It is obviously in the organization's best interest to include as many aspects of savings as possible. For the calculation of cost, however, this option does not exist; the organization should determine and include all costs.

Example 4 illustrates how an organization can apply the SPI benefit index method to determine the overall benefit from its software process improvement effort.

Example 4: Using the four projects from Acme Software in Examples 2 and 3: what is the total savings received by Acme Software for its improvement activities? In this example, we only have savings calculated from the reduction of errors and a decreasing level of effort needed to complete an equal amount of work. Equation 3 for Acme Software's improvement effort becomes:

SPI Savings = Σ[savings from(making fewer errors + decreasing warranty cost and/or maintenance)]

Recall that the calculated savings were:

Acme Software savings ($) from the reduction of errors

Year	Project A	Project B	Project C	Project D
1985-86	1,005,000	—	—	360,000
1986-87	637,800*	799,800*	—	263,400
1987-88	—	940,200	679,800	—
Total	1,642,800*	1,740,000*	679,800	623,400

* based on projected costs for a complete year

## Acme Software savings ($) from decreased level of effort

Year	Project A	Project B	Project C	Project D
1985-86	277,850	—	—	227,250
1986-87	414,800*	239,200*	—	57,450
1987-88	—	150,450	388,450	—
Total	692,650*	389,650*	388,450	284,700

* based on projected costs for a complete year

Using equation 3, we get a combined projected savings of $6,441,450 for the time period in question ($4,686,000 + $1,755,450).

Once the savings and costs are calculated using equations 2 and 3, then, using equation 4, the organization can calculate a data point for an indicator similar to the one in Figure 9. From Figure 9, the intent is obviously to have an index greater than one. Even though an index less than one implies that the SPI effort is costing more than it is saving, at least directly, an index less than one might be acceptable. Indirectly, other benefits might be seen in indicators not included in the savings equation, e.g., improvements in quality such as reliability or maintainability. To use Figure 9 for the overall benefit to the organization, the cost and savings data used to calculate the index would be normalized to a measure of software size.

Benefits that might be seen in other indicators (e.g., a maintainability indicator) are often difficult to attach a dollar value to and, as a result, are best tracked separately. It is also possible that benefits are being received that are not seen or are difficult to measure, e.g., new revenues, repeat business, or increased customer satisfaction. It is expected that benefits, measured or otherwise, will lag the actual improvements. In any case, the SPI benefit index should be analyzed concurrently with indicators not included in the savings portion of its equation to determine the overall benefit received from the SPI effort.
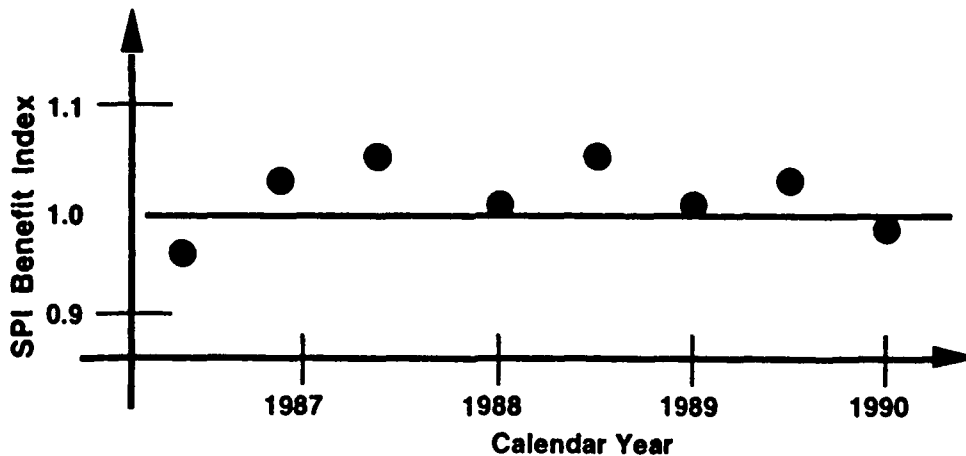
Figure 9 - Sample SPI Benefit Index Tracking

When analyzing the SPI index to determine the benefit, the organization also should look at the numerator and denominator, not just the index. The organization would compare the denominator, i.e., the dollars spent on software process improvement, to the organization's total budget for software development and maintenance, because the easiest way to drive the index up is to drive the cost down. When that happens, organizations are usually cheating themselves out of larger potential gains.

4.2 Summation of Benefits Method

Another method for determining the overall benefits derived from a SPI effort is to calculate the costs and savings for each improvement activity or change made as a result of the effort. For each change, x, the benefit equation now becomes:

$$\text{SPI Change Benefit (x)} = \$ \text{ saved (x)} - \$ \text{ cost (x)} \qquad \text{(eq. 5)}$$

The benefit that the organization receives is then the summation of the change benefits. From this we subtract the nonrecurring costs and add any benefit that can be determined from new or increased sales or business. Using this method, the equation for determining the overall benefit to the organization from the SPI effort then becomes:

$$\text{Total SPI Benefit} = \sum(\text{change benefits}) - \text{nonrecurring costs} \qquad \text{(eq. 6)}$$

This method might seem easier to apply, but there are advantages and disadvantages in its use. An advantage is that it may be easier to determine overlapping costs; however, there is an equal disadvantage because the accounting of the savings for each change makes it more difficult to determine overlapping savings. Another advantage is that the method can highlight the changes that have made the greatest impact, both in terms of total dollars saved and as a relative ratio of individual savings and costs of each change. However, those activities that are operating at a loss may be needed to optimize the overall process. The disadvantage is that there will be a temptation to eliminate those activities, thereby upsetting the balance and the system.

Another disadvantage is that, because a single dollar value is shown for the effort, people will tend to concentrate on the dollar value as the bottom line. But as in the SPI benefit index method, a negative total benefit using only dollars saved and costs might not be as bad as it looks if other improvement indicators are showing positive results and increasing the total dollars returned. Eventually, improved reliability and customer satisfaction indexes will make a positive impact on the organization's bottom line.

4.3 Considerations When Calculating Costs and Savings

When considering savings (and costs), an organization should ensure that the values are not counted twice. For example, an organization can determine a dollar amount saved by increasing its productivity. That organization can also determine a dollar value for reducing its errors or for finding errors earlier in the process, making them less costly to correct. It is probable that improvements in reducing total errors or finding errors earlier in the process are directly related to improvements in productivity. The organization should not count this savings twice. Separating the savings, though, may be difficult. A flow chart of the organization's processes showing the inputs and outputs of each process can help identify and separate items that might be counted twice.

Another cost that an organization may mistakenly count twice is the cost of implementing a process that is needed to compete in the marketplace versus a process implemented to improve an existing process in the organization. An example might be changes the organization makes to receive certification under the ISO 9000 series of international standards. In such cases, the organization should consciously decide whether or not to include those changes as part of its improvement efforts. If the changes are included, it may be helpful to track the cost and savings separately, where possible.

5. Other Measures for Determining Improvement

Sometimes it is difficult to measure the benefit received from a SPI effort in dollars, mostly because the benefit received is not dollars saved, but rather, is an expansion of gross income. It becomes difficult to determine whether the benefit is from the SPI effort or if marketing is doing a better job, the organization has found a niche for its product line, or if one of many other possible scenarios is occurring. Often, the benefit results from the company producing a higher quality product and, eventually, customers migrating to the better quality.

The difficulty is determining the benefit of new or more revenues returned by improving quality. Sometimes, an organization can only hope that, if it produces a better quality product, customers will buy the product at an increased cost. As ALCOA has learned, though, "it's a tough sales job" [SCHROEDER].

This chapter discusses some basic ways to measure how an organization's quality is improving. This chapter does not attempt to put a dollar value on benefits described, but if the organization can, it should. The organization would then include the value in the dollars saved portion of the SPI benefits index (eq. 4) or the savings portion of the summation of benefits method (eq. 6). The aspects of quality to be discussed here are:

- Mean time between failures. Here, we are interested in estimating when the product will fail.

- Mean time to repair. Many customers do not care what went wrong; they want to know when they can expect to use their system again.

- Availability, as determined by a combination of the mean time between failures and the mean time to repair. Again, customers are concerned about having the system operable when they need it.

- Customer satisfaction, probably the most overlooked measure of quality. The organization needs to ask the question: Are our customers satisfied with our products?

5.1 Mean Time Between Failures

Reliability modeling is a complex subject and a literature search will uncover many items on the subject. For the purpose of this report, the intent is not to provide a lengthy discussion on modeling reliability, but to introduce the reader to the subject of measuring the quality of a product by using the product's mean time between failures (MTBF).

The MTBF of a product is usually calculated by one of three basic methods. Each of the methods predicts the number of residual errors in the system to determine the MTBF of the software [LEONE]:

- Error correlation model: implements the viewpoint of reliability based on the number of errors corrected.

- Error detection model: uses elapsed testing time and the number of errors detected as the basis for the prediction.

- Simulation of the seeding of errors where errors are "seeded" into the system: Compares the ratio of seeded errors found during testing and the number of errors seeded to the number of "unseeded" errors found. This ratio is then used to predict the number of errors remaining.

The intent of these models is to be able to illustrate the MTBF of the organization's products similar to Figure 10. Ideally, the organization is able to determine a cost savings using an earlier technique and attach a dollar value to the benefit of improving its MTBF by some factor.
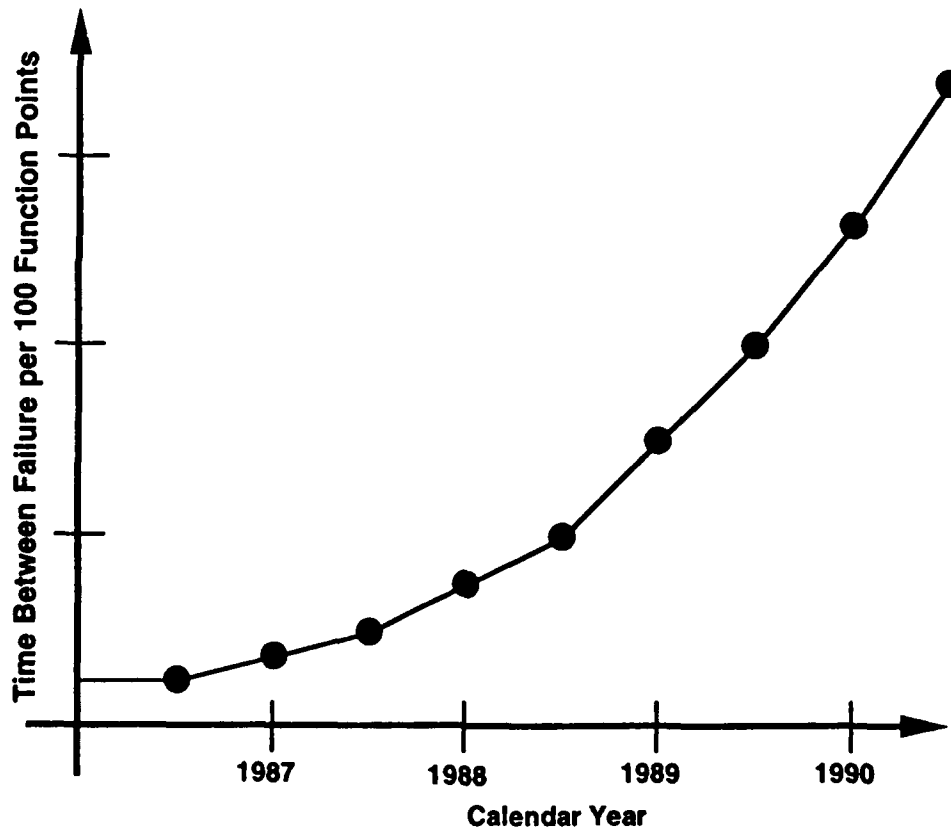

Figure 10 - Sample MTBF for an Organization's Products

5.2 Mean Time to Repair

Unlike the MTBF, which uses primarily testing data to predict its reliability, the mean time to repair (MTTR) is calculated using operational data. MTTR data usually come from data collected from service calls or warranty work. Here, an organization is trying to reduce its MTTR. An organization illustrates the changes in its MTTR using an indicator similar to Figure 11.
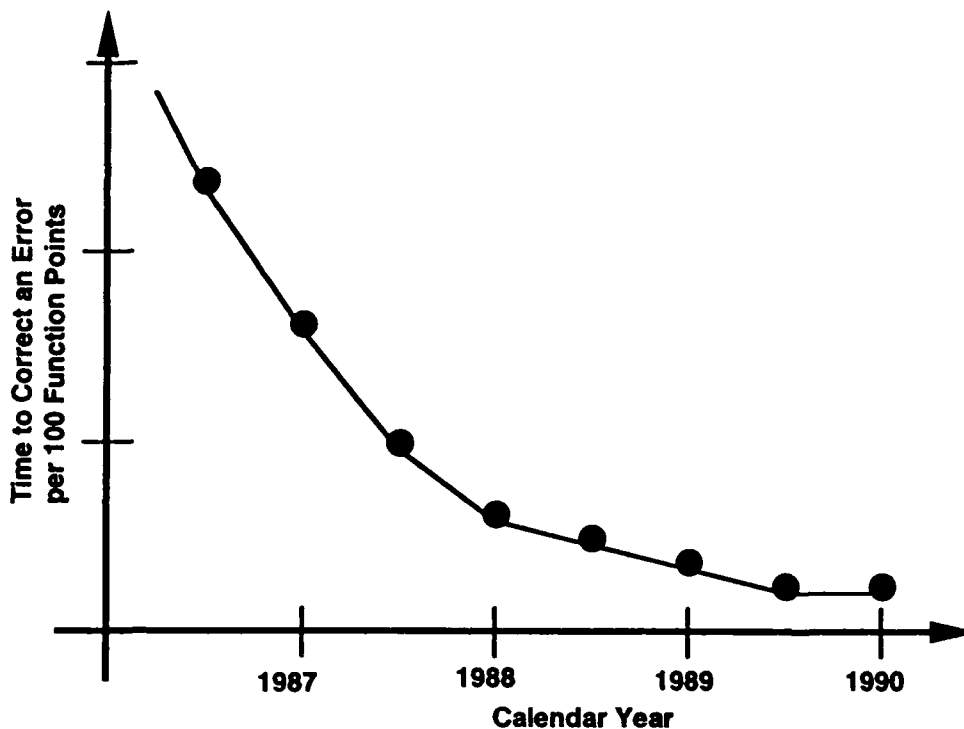

Figure 11 - Sample MTTR for an Organization's Products

The time to repair is calculated as a unit of time (days, hours, etc.) from when the organization is notified that a failure has occurred until the system is repaired and restored to its full operational capability. The MTTR is then the total of all the repair times divided by the number of problems and then normalized with the organization's size measure. Also, unlike the MTBF data, the MTTR data can be easily used to calculate savings in dollars by attaching a dollar value to the change in MTTR. Usually, this is done by comparing the warranty cost or service repair cost of successive time periods.

5.3 Availability

Availability is the percent of time the product is available during the time the product is expected to be operational. There are two types of availability, predicted and actual.

Predicted availability is calculated using the MTBF along with the MTTR and is the predicted future availability of the product. This type of availability is calculated as in eq. 7

$$\text{Predicted Availability} = \left(1 - \frac{\text{MTTR}}{\text{MTTR} + \text{MTBF}} \right) \times 100 \qquad \text{eq. 7}$$

Predicted availability can be calculated using data from operational testing. After the product is released and in use, the predicted availability is then forecast using operational data.

Actual availability is simply the percent of time the product is available for use during the time the product is expected to be operational. Down time for scheduled maintenance is usually not counted against the product's availability.

Of course, the goal for the organization is to try to increase the availability of its products. However, it is difficult to attach a dollar value to the change in availability. Often, this benefit shows up in increased sales, decreased follow-on work, and improved customer satisfaction.

5.4 Customer Satisfaction

Everybody who provides a product or service strives to achieve satisfied customers. However, sometimes it is difficult to determine when customers are satisfied. At the other end of the spectrum, it is easy to determine when a customer is not satisfied. The problem becomes: How does an organization measure customer satisfaction? Two possible measures are simply to measure the gross income and the number of customers and assume that trends in these measures also reflect the satisfaction of customers. These measures will provide data on the growth (or lack of growth) that the organization is experiencing. Neither measure, though, is easy to use to determine the direct benefits received from an organization's SPI effort and both could be heavily influenced by other market conditions, e.g., the amount and quality of the competition.

The only way to determine customer satisfaction is to ask customers if they are satisfied, what they like, what they would like to change, etc. This will require a customer survey and a process for applying it [CASSELL]. Developing a process for administering and tracking customer satisfaction surveys can be quite complex and is probably a good job for a consultant skilled in developing such surveys and questionnaires. Getting customers to complete the questionnaires might require direct personal contact by the organization. Certainly, follow-up feedback to customers who return a survey that rates the organization poorly should be direct, personal contact, and preferably by an upper level manager within the organization.

After the process is in place, the organization can develop a customer satisfaction index. This index would then be tracked monthly. Using the customer satisfaction index, data on the number of customers, and the gross income for the organization, the organization might be able to link the changes and determine a dollar value for changes in its customer satisfaction index. This dollar value would then be added into the savings portion of the SPI benefit index equation.

6. Summary and Conclusion

An organization should start to measure the benefits of its improvement efforts by first establishing a baseline using processes already or partially in place. The organization's upper management should establish a policy that makes each of these processes a requirement for all projects. As the level of conformance to the policy increases and the variance in the data decreases, the organization can begin to identify other methods for measuring its costs and savings, if necessary. If other measures are identified and processes are defined to implement them, the organization should pilot test the measures and processes on a portion of its organization prior to broad implementation throughout the organization.

As the organization begins to define and implement processes similar to level 2 processes of the SEI Capability Maturity Model [PAULK] [HUMPHREY], data items that measure the variability over time should become natural artifacts of those processes.

An organization should not attempt any of the recommendations on expanding the initial set until after (1) it has processes in place to collect and report the initial data items and has confidence in their validity and consistency, and (2) its practice includes regular reporting of the data items.

There are many ways to apply software measurement to determine the benefits an organization gains from its software process improvement efforts. Likewise, one measure or index will typically not be able to show the overall change and benefit an organization receives from its software process improvement activities. Nor does this report try to cover all the possibilities. Another source of measures that contains information and methods that can be adapted and applied is [BAUMERT]. The key is to choose a measurement method that includes a set of measures, establish a baseline for the current software processes and measures of its performance, and, using the measurement method, determine the changes to the baseline as improvements are made.

Appendix A — Acronyms

CDRL	Contract data requirements list
CMM	Capability Maturity Model
CY	Calendar year
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Points User Group
ISO	International Standards Organization
MTBF	Mean time between failures
MTTR	Mean time to repair
PDF	Probability density function
ROI	Return on investment
SEI	Software Engineering Institute
SEPG	Software engineering process group
SPI	Software process improvement
SSC	Standard Systems Center
WBS	Work breakdown structure

Bibliography

[BAUMERT]   Baumert, J. H. and M. S. McWhinney, *Software Measures and the Capability Maturity Model* (CMU/SEI-92-TR-25, DTIC ADA 258 238). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1992.

[BOEHM]   Boehm, B. W., *Software Engineering Economics*, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1981.

[CASSELL]   Cassell, R. H., "Seven Steps to a Successful Customer Survey," *Quality Progress*, July 1992.

[CONWAY]   Conway, W. E., "Quality Management in an Economic Downturn," *Quality Progress*, May 1992.

[CURTIS]   Curtis, B., "Software Process Improvement Seminar for Senior Executives," Software Engineering Institute Educational Series, 1992.

[DION92a]   Dion, R., "Elements of a Process Improvement Pr... ;-n," *IEEE Software*, July 1992.

[DION92b]   Dion, R., "Cost of Quality as a Measure of Process Improvement," Software Engineering Institute Software Engineering Symposium Proceedings, September 1992.

[FLORAC]   Florac, W. A., *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22, DTIC ADA 258 556). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1992.

[GOETHERT]   Goethert, W. B., E. K. Bailey, and M. B. Busby, *Software Effort and Schedule Measurement: A Framework for Counting Staff-Hours, and Reporting Schedule Information* (CMU/SEI-92-TR-21, DTIC ADA 258 279). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1992.

[HALSTEAD]   Halstead, M. H., *Elements of Software Science*, New York, NY: Elsevier Publishing, 1977.

[HUMPHREY]   Humphrey, W. S., *Managing the Software Process*, Reading, Massachusetts: Addison-Wesley Publishing Co., 1989.

[IFPUG90]   *Function Points As Assets: Reporting To Management*, International Function Points Users Group, 1990.

[IFPUG92]   *Function Point Counting Practices Manual*, Release 3.4, International Function Point Users Group, 1992.

[JONES]   Jones, T. C., *Applied Software Measurement-Assuring Productivity and Quality*, New York, NY: McGraw-Hill, Inc., 1991.

[LEONE]   Leone, A. M., *Selecting an Appropriate Model for Software Reliability*, 1988 Proceedings IEEE Annual Reliability and Maintainability Symposium, 1988.

[PARK]   Park, R. E., *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20, DTIC ADA 258 304). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, September 1992.

[PAULK]   Paulk, M. C., B. Curtis, and M. B. Chrissis, *Capability Maturity Model for Software, Version 1.1* (CMU/SEI-93-TR-24, Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, February 1993.

[PUTNAM90]   Putnam, L. H., "Is There a Real Measure for Software Productivity," *Programmer's Update*, June 1990.

[PUTNAM92]   Putnam, L. H. and W. Myers, *Measures for Excellence: Reliable Software On Time, Within Budget*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc. 1992.

[ROZUM]   Rozum, J. A., *Software Measurement Concepts for Acquisition Program Managers* (CMU/SEI-92-TR-11, DTIC ADA 258 177). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, June 1992.

[SCHROEDER]   Schroeder, M. and Z. Schiller, "Has ALCOA Found A Way to Foil the Aluminum Cycle," *Business Week*, January 8, 1990.

[WILLIS]   Willis, R. R., "Lessons Learned In Software Process Improvement," *Strategic Software Systems Conference*, Huntsville, Alabama, 1992.

[SNYDER]   Snyder, T. R., R. R. Willis, and W. S. Humphrey, "Software Process Improvement at Hughes Aircraft," *IEEE Software*, July 1991.

[WINSTON]   Winston, W. L., *Operations Research: Applications and Algorithms*, Boston, Massachusetts: Duxbury Press, 1987.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None			
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-93-TR-09		5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-93-186			
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute	6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office			
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213		7b. ADDRESS (city, state, and zip code) ESC/ENS Hanscom Air Force Base, MA 01731-2116			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SEI Joint Program Office	8b. OFFICE SYMBOL (if applicable) ESC/ENS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003			
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213		10. SOURCE OF FUNDING NOS.			

		PROGRAM ELEMENT NO 63756E	PROJECT NO. N/A	TASK NO N/A	WORK UNIT NO. N/A

11. TITLE (Include Security Classification)

Concepts on Measuring the Benefits of Software Process Improvement

12. PERSONAL AUTHOR(S)

James A. Rozum

13a. TYPE OF REPORT Final	13b. TIME COVERED FROM TO	14. DATE OF REPORT (year, month, day) June 1993	15. PAGE COUNT 52 pp.

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	capability maturity model process improvement software measurement return on investment process improvement benefits, costs, improvement savings

19. ABSTRACT (continue on reverse if necessary and identify by block number)

The software community initially became aware of process improvement from the works of Deming, Juran, and Crosby. The current awareness and activity regarding software process improvement was sparked by the Software Engineering Institute (SEI) with the release of its original software process maturity model. Following the advice of the SEI, many software organizations initiated software process improvement efforts to improve the quality of their products by improving the processes that produce those products. The question that many managers are continually asking today, though, is: How much has my organization benefited from the changes we have made? Unfortunately, many organizations did not include a method of measuring those benefits in their improvement activities. This report describes some concepts that organizations can tailor and build upon to develop a method for determining the benefits they have received from their software process improvement activities.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution	
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631	22c. OFFICE SYMBOL ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19

ABSTRACT — continued from page one, block 19